TÉCNICAS DE AUTO ESCALADO DE CLOUD COMPUTING APLICADAS AL ESTEGANOANÁLISIS

IÑIGO SAN ANICETO ORBEGOZO

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTESNE DE MADRID

Trabajo Fin Máster en Cloud Computing aplicado a Esteganografía

Septiembre 2011

Directores y colaborador:

Rafael Moreno Vozmediano
Ruben Santiago Montero

Autorización de difusión

Iñigo San Aniceto Orbegozo

07/09/2011

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "TÉCNICAS DE AUTO ESCALADO DE CLOUD COMPUTING APLICADAS AL ESTEGANOANÁLISIS", realizado durante el curso académico 2010-2011 bajo la dirección de Rafael Moreno-Vozmediano y Ruben Santiago Montero en el Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen en castellano

En este Proyecto Final de Máster se estudia un sistema de esteganografía y un algoritmo de esteganoanálisis para romper este sistema. Tras desarrollar el algoritmo de esteganoanálisis, este se adapta para poder ejecutarse en una arquitectura de cloud computing auto-escalable y se estudian las ventajas de romper así el sistema de esteganografiado.

Palabras clave

Esteganografía; Esteganoanálisis, Cloud Computing, Auto-Escalabilidad

Abstract

In this Master Thesis, a steganography system is analysed and, an algorithm to broke the system is presented. After developing the steganoanalysis algorithm, the algorithm is adapted to be executed in an auto-scalable cloud computing architecture and the advantages of breaking the system using this method are studied.

Keywords

Esteganography; Esteganoanalisis, Cloud Computing, Auto-Scalability

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Desde la antigüedad la necesidad de mantener ocultas las comunicaciones ante posibles intrusos ha estado presente en todas las civilizaciones.

Existen varios factores a tener en cuenta para mantener una comunicación lo más segura posible. Por un lado esta la seguridad del canal, esto es, que el intruso no pueda obtener el mensaje fácilmente del canal. En este contexto podríamos diferenciar canales confinados como en una comunicación por fibra óptica donde el intruso tiene que pinchar el cable en algún punto para obtener el mensaje y por lo tanto una vigilancia del cable nos protege contra una intrusión y canales no confinados como las comunicaciones por radio donde el intruso puede estar en cualquier sitio dentro del alcance de la onda de radio e interceptar el mensaje.

Hemos visto la seguridad del canal con respecto a la capacidad del intruso de obtener el mensaje y la diferencia a este respecto de los canales confinados y no confinados. Ahora, nos ponemos en la suposición pesimista de que el enemigo ya ha obtenido el mensaje. Existen diversas técnicas para protegernos eneste escenario. Por un lado podemos cifrar el mensaje. Esto es, podemos hacer una serie de transformaciones en el mensaje, de forma que, aunque el intruso lea el mensaje, este resulte incomprensible. Por otro lado podemos ocultar el mensaje. Esta es la técnica que vamos a considerar en este trabajo. La idea principal de esta técnica es que aunque el intruso obtenga acceso a la comunicación, el mensaje importante vaya oculto dentro de otro mensaje sin importancia o incluso contradictorio al mensaje importante, de forma que el intruso interprete el mensaje de forma errónea al no acceder al mensaje oculto.

Veamos esto con un sencillo ejemplo, enviamos un mensaje como el siguiente (Los espacios en blanco se representan con guiones): "Todo–va–bien.–No—os—preocupéis—mañana–volvemos–a–casa."Si un intruso intercepta esta señal pensaría que todo va bien, sin embargo, si analizamos los espacios entre caracteres y asignamos un punto a dos espacios en blanco y una raya a tres espacios en blanco tenemos (...–...) que en código Morse significa SOS esto es un mensaje contradictorio con el mensaje anfitrión. Este es un ejemplo muy sencillo pero ilustra la muy bien la idea de la esteganografía.

Obviamente, la comunicación más segura es aquella que evita que el intruso acceda al mensaje, y que este vaya oculto y encriptado para que en el escenario pesimista de una

intrusión el mensaje se transmita con seguridad.

1.1. Introducción a la Esteganografía: Ocultación del mensaje

Como ya se ha explicado, la técnica de la esteganografía consiste en ocultar un mensaje llamado mensaje huésped en otro llamado mensaje anfitrión. Este proceso depende de la naturaleza del mensaje huésped y anfitrión (texto, imagen, audio,vídeo) puesto que el proceso que permite ocultar uno en otro cambiara para cada uno de los casos.

Debido a las implicaciones que tiene la esteganografía en la seguridad de las comunicaciones y al creciente temor por la seguridad en la red la esteganografía ha sido un tema muy estudiado en la literatura[43], [39], [42], [44], [25], [45], [37], [36], [20], [19].

1.1.1. Objetivo de la Esteganografía

El objetivo final de la esteganografía es ocultar el mensaje huésped en el anfitrión de tal manera que el mensaje huésped resulte indetectable. Esto dependerá de muchos factores: tamaño del mensaje huésped y anfitrión, técnica utilizada, estadísticas del mensaje huésped y anfitrión, etc. En general, cuanto mayor sea el mensaje anfitrión y menor el mensaje huésped más fácil sera ocultar un mensaje, sin alterar significativamente las estadísticas del mensaje anfitrión siendo el mensaje huésped más difícil de detectar.

Existe una diferencia fundamental entre la esteganografía y la criptografía que conviene resaltar. La esteganografía trata de ocultar el mensaje mientras que la criptografía trata de hacerlo ilegible. Obsérvese que ambas técnicas son complementarias puesto que se puede hacer ilegible y después ocultar este mensaje cifrado para hacer aún más difícil obtener el mensaje original. En este trabajo sin embargo, tan solo nos centraremos en la esteganografía.

Un ejemplo típico en el que se aplica la esteganografía es en técnicas de watermarking o firma digital que sirven para asegurar la propiedad intelectual ante posibles robos[29], [31]. Por ejemplo: Creamos un vídeo que vamos a colgar en Internet pero no queremos que nadie lo use con fines lucrativos. Si en un futuro descubrimos que ese vídeo se ha usado en algún otro lado, nosotros podríamos demostrar la propiedad intelectual obteniendo la firma digital o el watermarking introducido por medio de técnicas esteganográficas.

1.1.2. Objetivo del Esteganoanálisis

Hasta ahora hemos visto todo desde el punto de vista de la persona que quiere ocultar su mensaje ante un posible intruso, pero es interesante también ver las cosas desde el punto de vista del intruso. En este caso, nuestro interés se centra en detectar posibles mensajes ocultos en las comunicaciones que estamos controlando. Habría que dar tres pasos: el primero sería obtener los mensajes del canal y posteriormente detectar los mensajes ocultos y finalmente descifrarlos.

En este trabajo solo nos centraremos en el segundo paso que es lo que se denomina esteganoanálisis. El esteganoanálisis requiere una gran cantidad de computo puesto que hay que probar para cada mensaje capturado muchas técnicas distintas de esteganografía y comparar las estadísticas obtenidas para determinar si hay o no hay o, al menos, si es susceptible de haber algún tipo de mensaje huésped dentro del mensaje anfitrión[29], [35], [28], [39], [42].

Estás técnicas pueden paralelizarse hasta un gran número de tareas independientes y, por tanto, pueden usarse técnicas de computación distribuida para reducir los tiempos de computo.

1.2. Introducción al Cloud Computing

El cloud computing es un nuevo paradigma de computación que pretende ofrecer la capacidad de computo como servicio. Tanto es así que algunas fuentes ya hablan del cloud computing como el quinto servicio[27] tras el agua, la electricidad, el gas y la telefonía. Esto supone un gran avance porque de ahora en adelante la capacidad instalada no es necesaria. Además el pago bajo demanda permite que el coste de una máquina durante 1000 horas cueste lo mismo que 1000 máquinas durante una hora abriendo la vía de la paralelización masiva de programas. Otra de las ventajas de que la capacidad de computo se ofrezca como servicio es la posibilidad de que los recursos utilizados varíen en el tiempo sin penalización por infrautilizar las infraestructuras. Esto es lo que se denomina por escalar los recursos. En cloud computing el administrador de un servicio puede aumentar o reducir los recursos fácilmente arrancando nuevas máquinas o apagando las ya arrancadas en tiempo de ejecución. Cambiando en tiempo real los recursos disponibles. Recientemente este concepto se ha ampliado a lo que se denomina auto-escalado. El auto-escalado es un escalado de recursos automático donde el administrador no actúa sobre los recursos disponibles manualmente. El papel del administrador en el auto-escalado es definir una serie de condiciones como carga de CPU, RAM, coste del servicio, etc. que al cumplirse lanzan disparadores para arrancar o apagar máquinas.

El auto-escalado puede ser gestionado por el proveedor cloud o por la aplicación ejecutada.

El esteganoanálisis representa un problema escalable con el número de mensajes a analizar y técnicas conocidas de esteganografía que se puede beneficiar de la auto-escalabilidad del cloud computing.

1.3. Objetivos del Trabajo

El objetivo del trabajo es crear una arquitectura de cloud computing que auto-escale de forma inteligente para adaptarse en tiempo de ejecución a las características de las imágenes que se están analizando y así reducir los tiempos de computo manteniendo un control del coste y posibilitar el procesado de una gran cantidad de imágenes.

El objetivo del trabajo no es crear nuevas técnicas de estegoanálisis sino demostrar que

las ya existentes se pueden modificar para funcionar eficientemente en una arquitectura de cloud computing con auto-escalado inteligente.

Para desarrollar el trabajo se crea un sistema de esteganografía que oculta el capítulo I de "El Quijote" en una imagen y un algoritmo de esteganoanálisis que trata de hallar mensajes en imágenes. Éste último algoritmo es el que se modifica para que se pueda utilizar en la arquitectura de cloud computing con auto-escalado inteligente pero en principio podría modificarse cualquier otro algoritmo paralelizable. Estos algoritmos no tienen porque ser de esteganoanálisis, si son paralalelizables y requieren de una gran cantidad de datos o producen una gran cantidad de resultados periódicamente está técnica mejorará considerablemente su tiempo de ejecución.

1.4. Aportaciones del trabajo

Las principales aportaciones de este trabajo son la arquitectura de auto-escalado inteligente, la adaptación de un algoritmo a esta arquitectura y la predicción precisa del tiempo requerido para la finalización del algoritmo que se usa como dato de entrada para que el algoritmo de auto-escalado decida el número de instancias necesarias para ajustarse a la hora de finalización establecida por el usuario.

1.5. Proyecto

En este proyecto se crea en un primer paso un sistema de esteganografía que oculta (sin encriptar) un mensaje de texto en español antiguo codificado en ASCII dentro de una imagen y se realiza el proceso inverso para obtener el mensaje.

En un segundo paso, se crea un algoritmo de esteganoanálisis que prueba muchas técnicas de esteganografía y compara las estadísticas obtenidas con las estadísticas de los caracteres del español antiguo y del ingles codificado en ASCII y con los resultados muestra si el mensaje es susceptible de contener algún mensaje huésped y la sección del mensaje en la que el mensaje huésped podría encontrarse.

Finalmente, en un tercer paso, se modifica el algoritmo de esteganoanálisis para que sea auto-escalable usando cloud computing y se muestran las mejoras obtenidas con respecto al algoritmo que trabaja en modo local.

1.5.1. Estructura del Proyecto

El resto del trabajo se desarrollara siempre de la siguiente forma: se presentará en primer lugar un estudio general sobre cada apartado (esteganografía, esteganoanálisis y auto-escalado en cloud computing) y posteriormente se explicará la parte del proyecto correspondiente a dicho apartado.

En el capítulo 2 se representa un esquema general de un sistema de esteganografía que incluye los estegomedios más comunes, los fines de la esteganografía, las herramientas comerciales y la parte del proyecto que trata sobre la esteganografía donde se presenta el

mensaje anfitrión, el mensaje huésped, el sistema de esteganografía que oculta y recupera el mensaje, el canal considerado y los cambios que se producen en las estadísticas de la imagen.

En el capítulo 3 se presentan y clasifican las técnicas de esteganoanálisis, se muestran herramientas comerciales y se presenta el algoritmo del proyecto. Este algoritmo trata de buscar mensajes ocultos dentro del mensaje anfitrión probando varias técnicas de esteganografía comparando las estadísticas de los resultados obtenidos y guardando el tiempo empleado en cada iteración para el análisis posterior.

En el capítulo 4 se estudia un sistema de cloud computing que auto-escala y se adapta a las necesidades de cálculo del esteganoanálisis para obtener el mejor resultado.

Finalmente en el capítulo 5 se presentan las conclusiones del trabajo y las líneas de trabajo futuro.

Con en fin de transmitir una visión general del proyecto en la Figura 1.1 se muestra la estructura completa del proyecto.

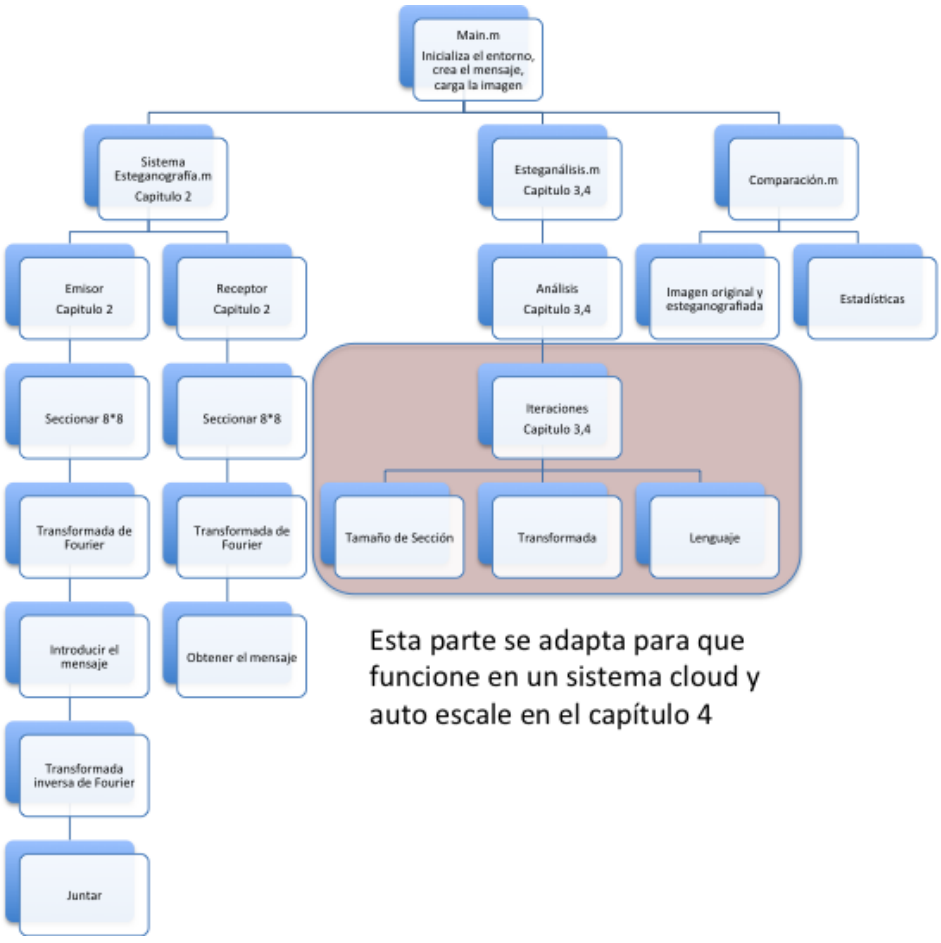

Figura 1.1: *Estructura del Proyecto*

Capítulo 2

Sistemas de esteganografía

2.1. Introducción

En este capítulo se explica en primer lugar en qué consiste la esteganografía presentando un modelo genérico de un sistema de esteganografía. Posteriormente, se explica la técnica de esteganografía que se utiliza en este trabajo para el posterior esteganoanálisis.

Para explicar la técnica de esteganografía utilizada en primer lugar se explica la clase de sistema esteganografico utilizado y posteriormente se detallan las características del mensaje anfitrión y del mensaje huésped. Finalmente se explica, paso por paso, el algoritmo de esteganografía y los cambios que produce en las estadísticas de los píxeles de la imagen.

2.2. Sistemas de esteganografía

En cualquier sistema de esteganografía tendremos un mensaje anfitrión, un mensaje huésped, un emisor y un receptor. Y en ciertos sistemas existirán además ciertas "llaves"que se usarán en el emisor y el receptor para ocultar y obtener el mensaje. En la Figura 2.1 podemos observar los componentes básicos de un sistema de esteganografía.

2.2.1. Características de un sistema de esteganografía

Un sistema esteganográfico se caracteriza por los siguientes parámetros[19]:

1. **Capacidad:** Es el número de bits que es capaz de esconder dentro de un mensaje anfitrión. Esta característica está relacionada con el sistema de esteganografía y con el mensaje anfitrión.

2. **Robustez:** La capacidad del sistema de someterse a transformaciones lineales y no lineales como filtros, adición de ruido, escalado, rotación, etc. sin perder el mensaje. Existen numerosas técnicas para añadir robustez al sistema. Algunas de ellas se analizan en[31].

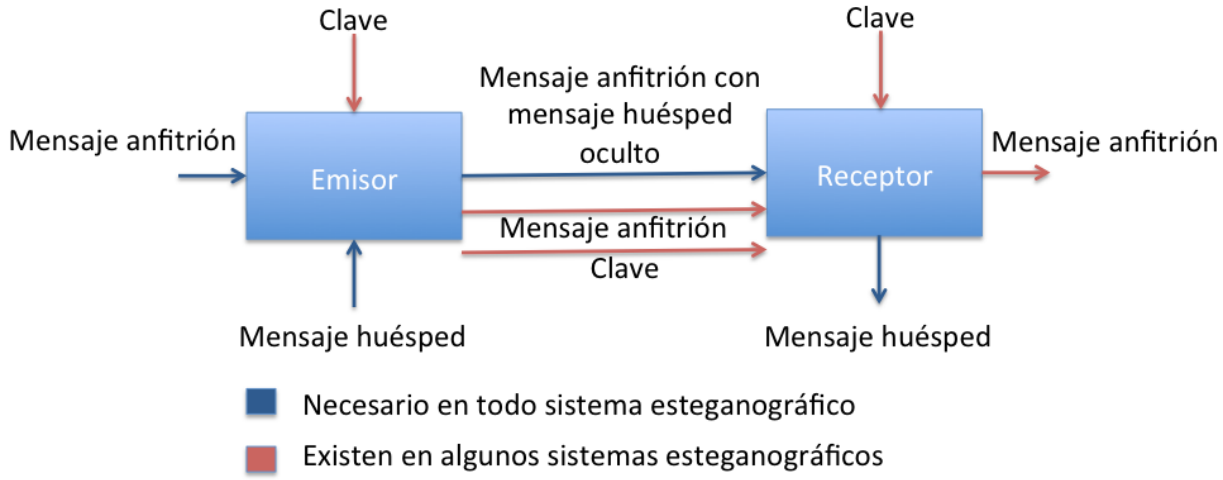

Figura 2.1: *Sistema de esteganografía*

3. **Detectabilidad:** Correlación del mensaje final respecto a la fuente que genera los mensajes. Esto es, que estadísticamente el mensaje generado sea o no consistente con la fuente que genera el mensaje anfitrión. Este parámetro está obviamente relacionado con el tamaño del mensaje a ocultar y el formato del mensaje anfitrión.

4. **Invisibilidad:** Es similar a la característica anterior pero está vez no está relacionado con las estadísticas sino que con la percepción humana. En general es más fácil hacer un sistema de esteganografía invisible que indetectable dadas las características de la percepción humana.

5. **Seguridad:** Esta característica mide la posibilidad de un atacante de borrar un mensaje una vez que ha descubierto que existe algo oculto. Existen diversas técnicas para borrar mensajes esteganográficos: añadir ruido, recortar, etc. Pero también hay técnicas para protegerse contra esto.

Al crear un sistema de esteganografía es muy importante tener en cuenta que una herramienta esteganográfica debe obtener un equilibrio entre la cantidad de información a ocultar y su detectabilidad y invisibilidad.

2.2.2. Clasificación de sistemas de esteganografía

Existen varios clasificaciones de sistemas esteganográficos en función de distintos factores[19]:

1. En función del tipo de mensaje anfitrión.

2. En función de la información transmitida entre el emisor y el receptor.

3. En función del tipo de modificación al que se somete el mensaje anfitrión.

En función del tipo de mensaje anfitrión

La primera clasificación es en función del tipo de mensaje anfitrión que usamos para transmitir. Tenemos por tanto sistemas para mensajes de vídeo, audio, imágenes...

En función de la información transmitida entre el emisor y el receptor

En esta clasificación tenemos[19]:

1. **Esteganografía pura:** En este sistema el mensaje transmitido consiste en un mensaje anfitrión con el mensaje huésped oculto. No existe ningún intercambio de llaves. El problema de este tipo de esteganografía es que no se obtiene ninguna seguridad si el atacante conoce el método de esteganografía usado.

2. **Esteganografía de clave privada:** Es similar al sistema anterior pero el mensaje se oculta usando una llave privada que proporciona más seguridad y que dado que tanto el emisor como el receptor conocen la llave no es necesaria su transmisión.

3. **Esteganografía de clave pública:** Normalmente se requieren dos claves una privada (secreta) y otra publica, la clave pública se usa para ocultar la información y la privada para reconstruirla. En este caso existe la necesidad de transmitir la llave a través del medio.

En función del tipo de modificación al que se somete el mensaje anfitrión

Es la clasificación que se usa normalmentey en ella se distinguen[19]:

1. **Sistemas que se basan en sustitución:** Se basan en sustituir bits insignificantes del mensaje anfitrión y recuperar el mensaje gracias al conocimiento de donde se encuentran estos bits. Dentro de esta clasificación encontramos: LSB (Least Significant Bit) que codifica el mensaje utilizando el bit menos significativo; Pseudorandom Permutation que codifica usando bits pseudo-aleatorios que dificultan el ataque al sistema; image downgrading que codifica una imagen dentro de otra codificando en los bits menos significativos los bits más significativos de la otra imagen; ”cover region and parity bits” que divide el mensaje anfitrión en secciones y guarda la información junto a bits de paridad para comprobar que no se ha visto alterado el mensaje por el ruido; etc.

2. **Sistemas basados en transformaciones:** El sistema de sustitución es altamente vulnerable ante posibles ataques, y aunque un atacante no halle el mensaje lo puede destruir fácilmente por medio de procesado digital de la señal ya que pequeños cambios pueden producir una completa pérdida de información. Se ha demostrado, que guardar información en el dominio de la frecuencia resulta mucho más robusto a este respecto[36].

La razón es que el mensaje huésped se guarda en una área significativa de la imagen por lo que añadir ruido, compresión o cortar la imagen no producen un efecto tan importante como el que ocurre en sistemas de sustitución donde la información se encuentra en una región muy pequeña. Entre los métodos utilizados están el Discrete Cosine Transformation (DCT), Fast Fourier Trasnformation (FFT), etc. La razón por la que estos sistemas son más robustos frente a ataques es que la mayoría de los ataques normalmente afectan a cierta banda de la transformada manteniendo el resto de los coeficientes prácticamente intactos.

3. **Sistemas de espectro ensanchado:** Se usa en medios de transmisión en los que el ancho de banda utilizado es mayor al mínimo necesario para mandar la información. Este sistema es similar a otros sistemas que tratan de difuminar el mensaje huésped en el mensaje anfitrión de forma que resulte muy difícil de percibir. Además, al estar difuminado resulta difícil eliminarlo. Las técnicas más utilizadas son las de secuenciación directa y las de salto de frecuencia. Estás técnicas son ampliamente utilizadas en sistemas de watermarking[31].

4. **Sistemas de distorsión:** La mayor diferencia de esta técnica con respecto a las anteriores es que resulta necesario conocer el mensaje anfitrión en el decodificador para ser capaz de obtener el mensaje. Este sistema realiza diversas modificaciones en el mensaje anfitrión para que introduzcan información del mensaje que transmite. El receptor compara las diferencias con el mensaje original y obtiene el mensaje huésped. La mayoría de los métodos para ocultar mensajes de texto se basan en este tipo de sistemas. Un ejemplo de este sistema es añadir espacios o caracteres "invisibles" al texto para mandar información adicional.

5. **Sistemas que generan un mensaje anfitrión:** Está técnica es radicalmente distinta a todas las anteriores. En vez de utilizar un mensaje anfitrión ya existente para ocultar el mensaje huésped lo que hace es generar un mensaje anfitrión con el único fin de ocultar en el un mensaje huésped.

Hemos visto las modificaciones más habituales para ocultar el mensaje, sin embargo, existen más tipos de modificaciones que permiten ocultar un mensaje lo que hace suponer que resultará muy complicado obtener el mensaje huésped si no conocemos el algoritmo utilizado.

Observase también que se pueden utilizar múltiples formas para ocultar el mensaje: En protocolos estándar de comunicación, en ficheros, etc.

2.2.3. Estegomedios

A la hora de transmitir un mensaje oculto es importante pensar en el canal. La primera idea en la que todo el mundo piensa son canales seguros donde la comunicación no pueda ser interferida por intrusos, sin embargo, hay otros medios que potencialmente resultan muy

seguros para este tipo de transmisiones[42]: Las redes sociales, páginas de vídeo/audio en streaming, etc.

En estos medios obtener el mensaje del medio resulta relativamente sencillo. Existen impedimentos: perfiles de privacidad, redes cerradas de amigos, autenticación, etc. pero son filtros de seguridad relativamente fáciles de saltar. Sin embargo, la cantidad de información existente y su movimiento es tal (upload /settings) que resulta muy complejo analizar todos los archivos (vídeo/audio/imágenes/texto) para separar archivos estenografiados de los que no. La Figura 2.2 ilustra la cantidad de información que se mueve en una red social. Además un solo mensaje se puede dividir y enviar por partes a través de múltiples redes sociales. Es por tanto, un medio idóneo para este tipo de comunicaciones. Además, cabe destacar que este tipo de comunicaciones está en constante expansión y por tanto, resulta cada vez más idóneo como se puede ver en la Figura 2.3.

Figura 2.2: *Esta imagen ilustra la cantidad de información y su movimiento en redes sociales*

2.2.4. Fines de la esteganografía

La esteganografía puede ser una herramienta beneficiosa para proteger la privacidad, derechos de autor, etc. Sin embargo, también puede usarse para fines ilegítimos como herramientas, que combinadas con estas técnicas, pueden incluso dañar la integridad de equipos informáticos con herramientas de hacking ocultas.

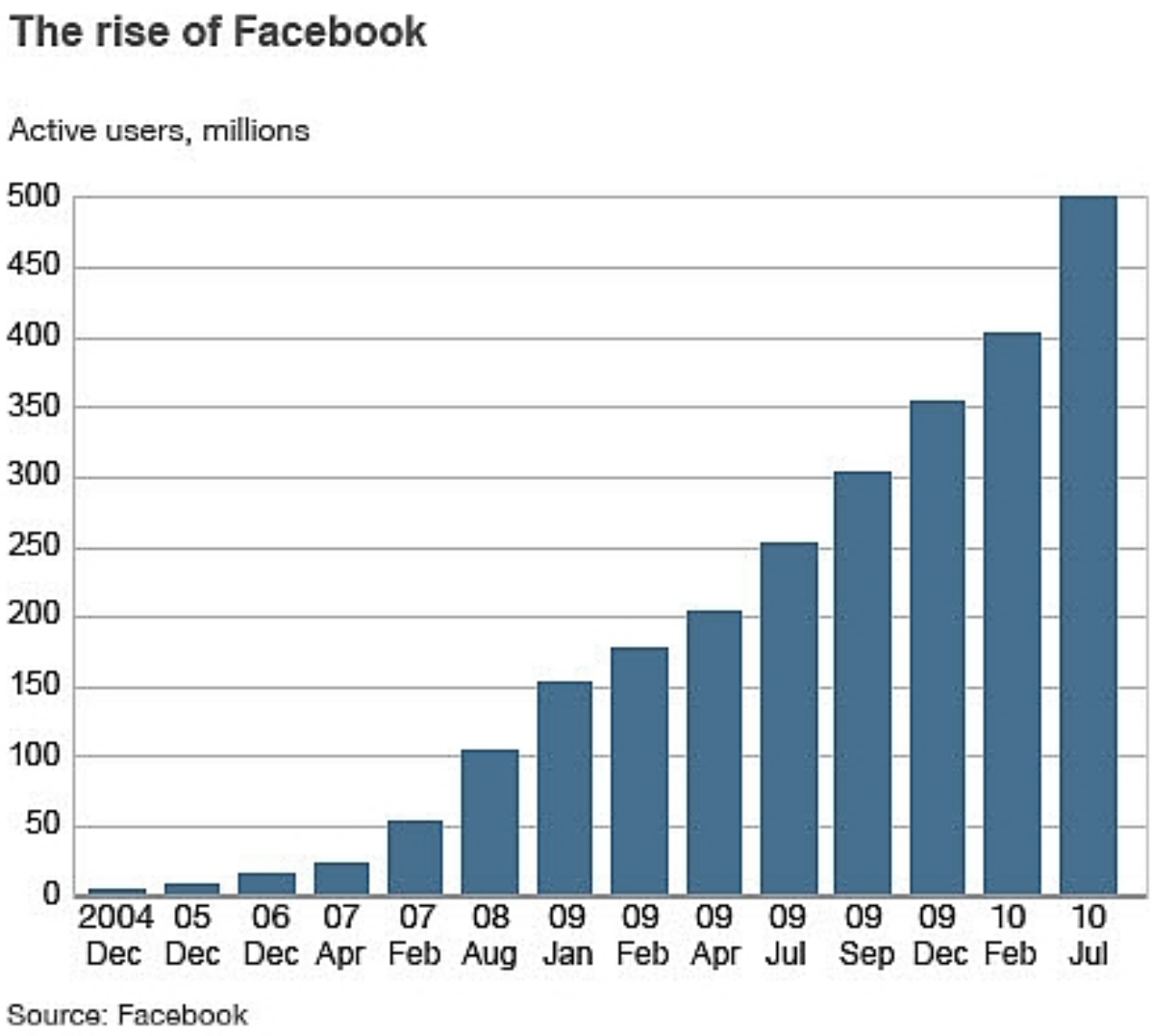

Figura 2.3: *Esta imagen ilustra la el crecimiento del número de usuarios en las redes sociales*

2.2.5. Herramientas comerciales

Existen numerosas herramientas comerciales de esteganografía, pero como veremos más adelante, no resultan muy útiles si queremos mantener una comunicación segura porque al conocerse el algoritmo de esteganografía es más sencillo hacer una herramienta de esteganoanálisis que analice estos métodos. Algunos ejemplos son: Jsteg, jphide, outguess, F5, appendX y Camouflage.

2.3. Sistema de esteganografía implementado

En este apartado se presenta el algoritmo de esteganografía que se ha desarrollado en este proyecto y se usará para el posterior esteganoanálisis. En primer lugar se presenta la clasificación y características de la técnica utilizada y posteriormente se detallan las características del mensaje anfitrión y huésped. Finalmente se analizan los cambios producidos en las estadísticas de los valores de los píxels y se detalla la implementación.

2.3.1. Técnica de esteganografía

La técnica de esteganografía empleada se basa en un sistema de esteganografía pura sin llave, basado en transformaciones. En concreto usa la Transformada Discretas de Fourier (DFT) añadido a un sistema de sustitución de los coeficientes obtenidos en el dominio de la frecuencia.

La técnica es relativamente sencilla: se coge la imagen del mensaje anfitrión, se divide por bloques y se aplica una transformada discreta de Fourier para obtener las componentes frecuenciales de cada uno de los bloques.

Una vez que tenemos las componentes frecuenciales codificamos usando el último bit que no se ve alterado por el canal (hay que guardar la imagen en números enteros del 0-255), parte del mensaje huésped codificado en ASCII y deshacemos la DFT para obtener otra vez el bloque. Repetimos este proceso hasta tener todo el mensaje oculto en la imagen.

Con el objetivo de mostrar la técnica, vamos a suponer un ejemplo con una imagen 24x16 píxels. En la Figura 2.4 viene explicado el proceso que se explicará a continuación.

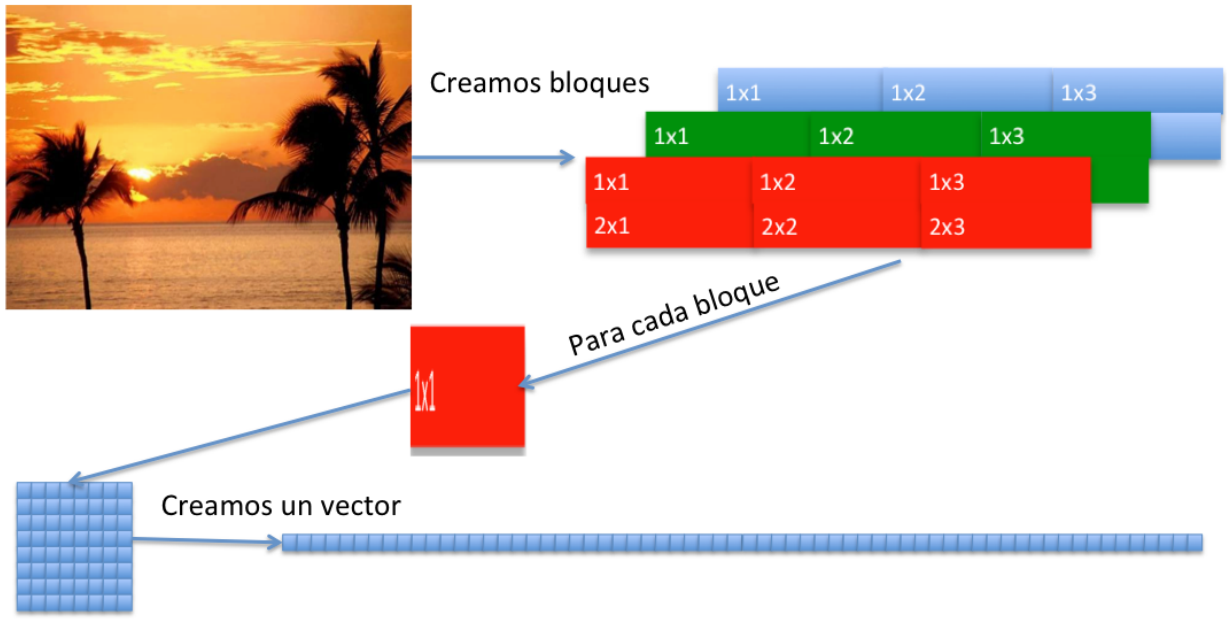

Figura 2.4: *Pasos a seguir antes de la Transformada Discreta de Fourier*

1. En primer lugar, dividimos la imagen en secciones de 8x8 (este tamaño se toma arbitrariamente, una técnica de esteganoanálisis tendrá que detectar el tamaño de sección usado). Obtendremos una hipermatriz de secciones de 2x3x3 (Alto, ancho, RGB).

2. En este punto, para cada sección, transformamos la matriz de 8x8 en un vector de 1x64 con el objetivo de aplicar la DFT (Podríamos haber usado otra transformada, como la del coseno, por lo tanto una técnica de esteganoanálisis también tendrá que detectar la transformada utilizada). En la Figura 2.5 podemos ver este vector. Tras aplicar la DFT a este vector obtenemos otro vector de 1x64 pero con valores complejos. Este resultado se muestra en la Figura 2.6:

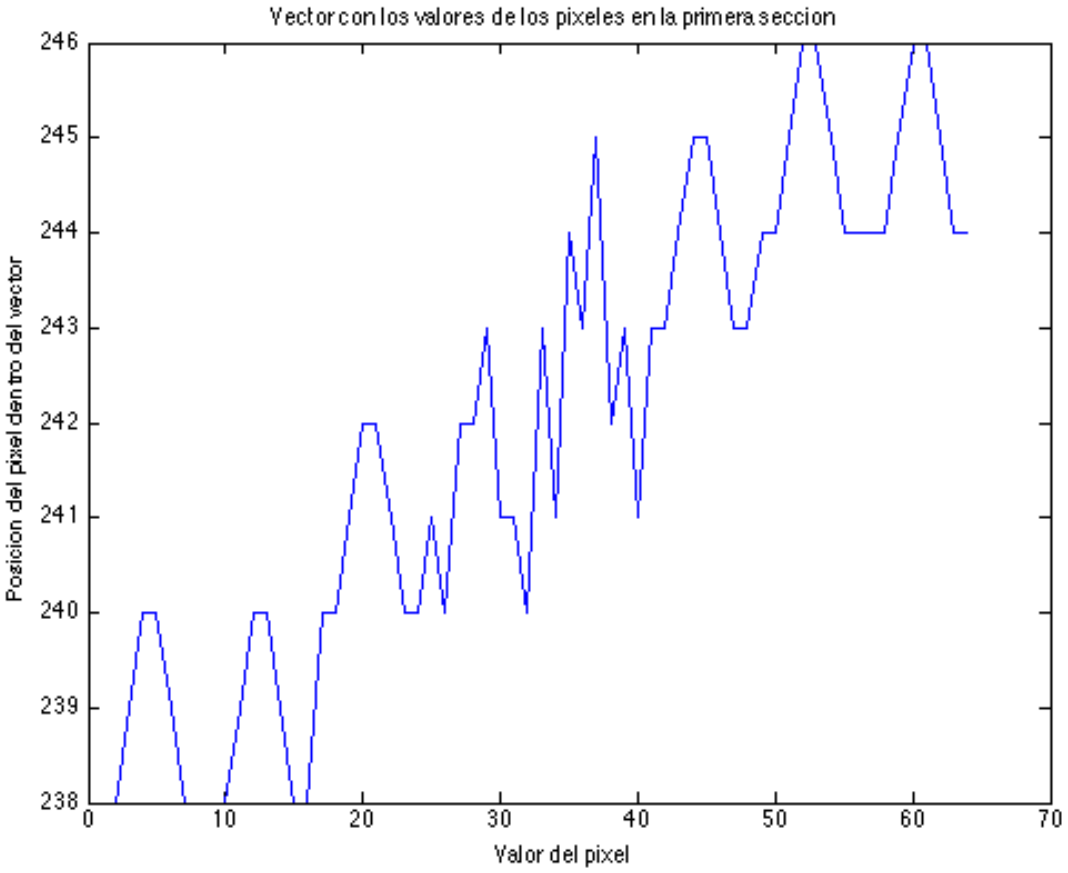

Figura 2.5: *Valores obtenidos en el vector 1x1x1 seccionado*

3. Ahora el paso a seguir es el siguiente, cogemos cada uno de los valores complejos y los transformamos al binario.

4. El bit menos significativo de la parte real del número complejo se utiliza para codificar un bit del carácter ASCII del mensaje huésped (Se podría codificar usando la

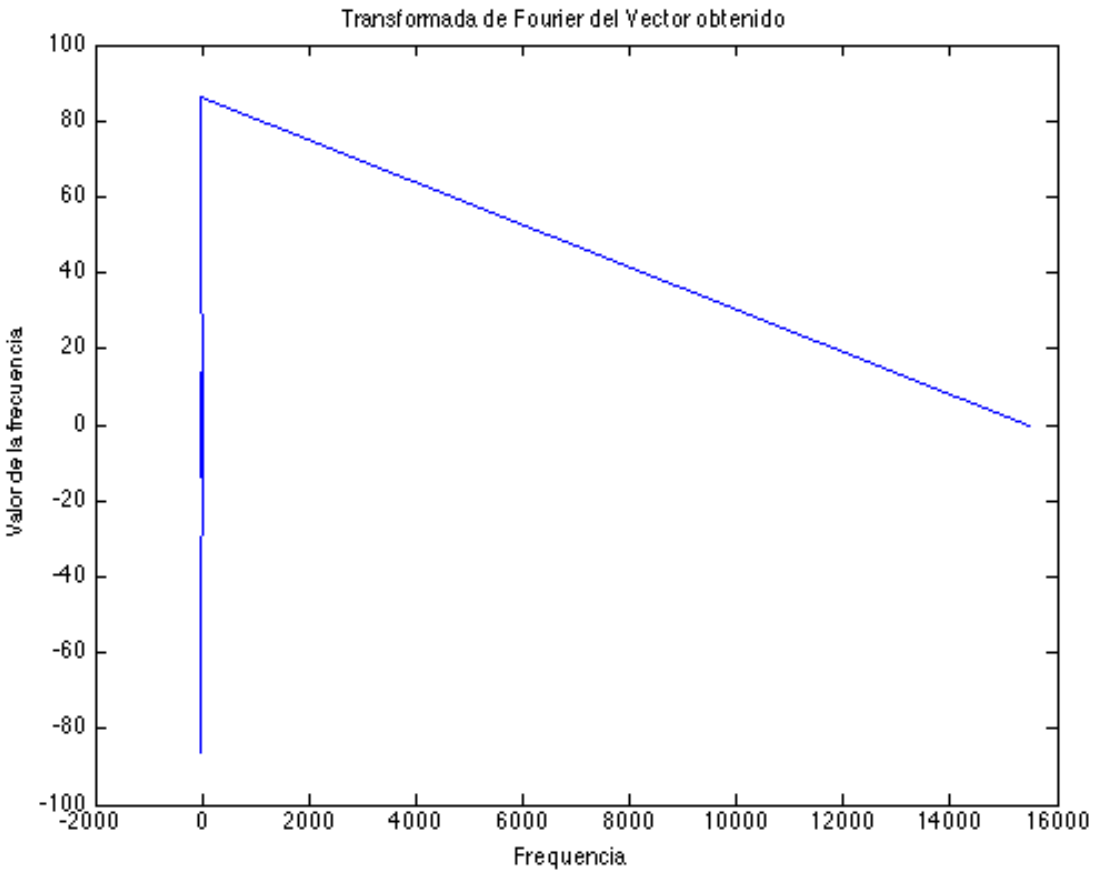

Figura 2.6: *Valores obtenidos en el vector 1x1x1 seccionado tras aplicar la DFT*

parte imaginaria, la magnitud, la fase, etc. por lo tanto, la técnica de esteganoanálisis también debe detectar esto).

5. Como necesitamos 7 bits para codificar un carácter ASCII necesitaremos siete valores de frecuencia para codificar cada carácter (Y como hemos visto anteriormente tenemos 64 valores en cada bloque por lo que también podremos elegir cual de ellos modificar). En la Tabla 2.1 vemos el valor de los coeficientes de la transformada discreta de Fourier para el primer bloque.

6. Al transformar a binario introducimos arbitrariamente un offset de 65536 para no tener números binarios negativos. Posteriormente eliminaremos este offset al volver a transformar el número a decimal.

7. El primer carácter que queremos meter es "C" que en ASCII es el número 67 y en binario **1000011**. En la siguiente Tabla 2.2 se muestran la parte real de los primeros siete valores complejos de la transformada discreta de Fourier en binario y como cambian estos valores tras la esteganografía.

Valor 1-16	Valor 17-32	Valor 33-48	Valor 49-64
n 1: 15485	n 17: 8+8i	n 33: 12	n 49: 8-8i
n 2: -12+86i	n 18: -5+2i	n 34: -16+1i	n 50: -5-5i
n 3: -8+38i	n 19: -5+3i	n 35: 4-2i	n 51: -3-4i
n 4: 4.70735e-14+21i	n 20: -3+1i	n 36: -8+3i	n 52: -5-3i
n 5: -2+7i	n 21: -1+1i	n 37: -1-2i	n 53: -1-2i
n 6: -7+12i	n 22: -5+1i	n 38: -2+1i	n 54: -3-5i
n 7: -3+11i	n 23: -2+1i	n 39: -7-2i	n 55: -5-7i
n 8: -4+12i	n 24: -6+2i	n 40: -3+8.5265e-14i	n 56: -5-8i
n 9: -33-14i	n 25: -1.0658e-13+2i	n 41: -1.0658e-13-2i	n 57: -33+14i
n 10: -5+8i	n 26: -3-8.5265e-14i	n 42: -6-2i	n 58: -4-12i
n 11: -5+7i	n 27: -7+2i	n 43: -2-1i	n 59: -3-11i
n 12: -3+5i	n 28: -2-1i	n 44: -5-1i	n 60: -7-12i
n 13: -1+2i	n 29: -1+2i	n 45: -1-1i	n 61: -2-7i
n 14: -5+3i	n 30: -8-3i	n 46: -3-1i	n 62: 4.70735e-14-21i
n 15: -3+4i	n 31: 4+2i	n 47: -5-3i	n 63: -8-38i
n 16: -5+5i	n 32: -16-1i	n 48: -5-2i	n 64: -12-86i

Cuadro 2.1: *Valores del vector tras la DFT*

Valor en binario 1-7	Valor en binario 1-7 tras esteganografía
10011110001111100	1001111000111110**1**
01111111111110101	0111111111111010**0**
01111111111111001	0111111111111100**0**
10000000000000000	1000000000000000**0**
01111111111111111	0111111111111111**0**
01111111111111001	0111111111111100**1**
01111111111111101	0111111111111110**1**

Cuadro 2.2: *7 primeras partes reales del vector transformada en binario antes y después de esteganografía.*

En este ejemplo de 24x16 píxels, tendríamos 2x3x3= 18 secciones y por lo tanto podríamos codificar 18 caracteres con este método. En la imagen que usamos para este trabajo tenemos 663x497 píxels, y por lo tanto 82x62x3=15252 caracteres.

2.3.2. Mensaje anfitrión

El mensaje anfitrión elegido para la esteganografía es una foto. Las características de la foto son las siguientes:

- **Tamaño de la foto:** 991KB. Recordemos que cuanto mayor es el mensaje anfitrión y menor el mensaje huésped más difícil resulta encontrar el mensaje huésped porque las estadísticas resultaran alteradas en menor medida.
- Resolución de la foto: 663x497
- **Código de colores:** RGB (Red Green Blue). Se codifica usando una matriz para cada uno de los colores. Se puede codificar usando 24 bits RGB(8,8,8) con 16,7 Millones de colores.
- **Codificación:** bmp (Bit Mapped Picture). En esta codificación cada píxel representa un valor dentro de una matriz.
- **Resumen:** La imagen se puede representar como una matriz de 663x497x3 con un valor de 0-255.

En la Figura 2.7 observamos el mensaje anfitrión antes de ser sometido al proceso de esteganografía.

2.3.3. Mensaje huésped

Como mensaje huésped hemos seleccionado una longitud de caracteres variable del primer capítulo del libro "El Quijote". Vamos a asumir una codificación ASCII (7bits) por lo que los caracteres: (á é í ó ú) serán sustituidos por (a e i o u) respectivamente.

A continuación presentamos el primer segmento del mensaje huésped.

CAPITULO 1: Que trata de la condición y ejercicio del famoso hidalgo D. Quijote de la Mancha

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivia un hidalgo de los de lanza en astillero, adarga antigua, rocin flaco y galgo corredor. Una olla de algo mas vaca que carnero, salpicon las mas noches, duelos y quebrantos los sabados, lentejas los viernes, algun palomino de anadidura los domingos, consumian las tres partes de su hacienda. etc

Figura 2.7: *Mensaje Anfitrión usado en el trabajo*

2.3.4. Cambios que se produce en las estadísticas de la imagen

Tras aplicar esta técnica con los primeros 3122 caracteres de "El quijote" el resultado que se obtiene es el siguiente.

Como podemos observar en la Figura 2.8 esta técnica de esteganografía resulta invisible puesto que aparentemente ambas imágenes son iguales. Tenemos que averiguar ahora si resulta indetectable.

Si analizamos las estadísticas de los valores de los píxeles de ambas imágenes tenemos que:

- La media del valor del píxel cambia y el valor medio del píxel pasa de 119,1256 a 119,1269 siendo el cambio del 0,00107 %.
- La mediana del valor de los píxeles de la imagen no cambia nada y se mantiene en 103.
- La varianza del valor de los píxeles de la imagen cambia y pasa de 7084.8248 a

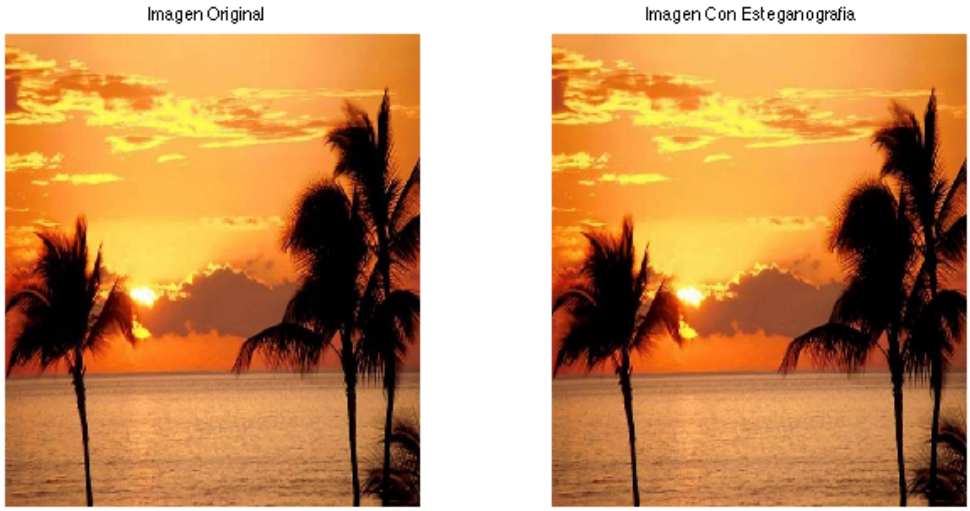

Figura 2.8: *Comparación de la imagen original con la esteganografiada*

7085.0402 siendo la diferencia del 0,0030408 % la característica estadística que más varía de la imagen original a la modificada.

- En la Figura 2.8 se muestra el histograma de los valores de los píxeles en la imagen original y en la modificada.
- Finalmente la desviación estándar del valor de los píxeles cambia y lo hace desde 84.1714 a 84.1727 siendo el cambio del 0,0015204 %.

Aparentemente las estadísticas varían muy poco y parece que sin tener ambas imágenes el mensaje resulta indetectable. Sin embargo, esto no es así. El problema es que estamos analizando las estadísticas equivocadas.

Cuando analizamos el histograma de los bits que se usan para modificar la imagen y lo comparamos con las estadísticas del idioma si observamos una pequeña anomalía: Los valores que se usan para codificar en ASCII aparecen con más frecuencia que el resto y parecen seguir un patrón similar al del idioma. En el siguiente capítulo veremos como tratar estos valores para crear un algoritmo de esteganoanálisis.

Esta última comparación que se muestra en la Figura 2.10 es la que se usa para detectar mensajes usando técnicas de esteganoanálisis comunes como se explicará en el siguiente capítulo.

2.3.5. Recuperar el mensaje

Conociendo el sistema utilizado para esconder el mensaje resulta muy sencillo volver a obtenerlo, realizando los mismos pasos hasta llegar a los números binarios.

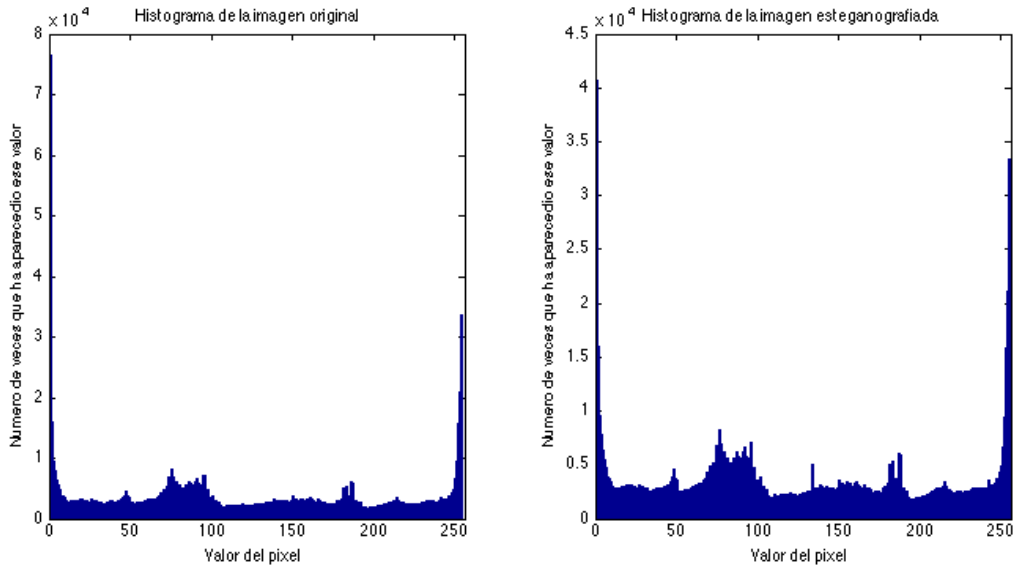

Figura 2.9: *Comparación del histograma de los valores de los píxeles de la imagen original con la esteganografiada*

1. Se secciona la imagen en bloques de 8x8 y se transforman en vectores de 1x64.
2. Se realiza una DFT a estos vectores.
3. Se coge la parte real, se le añade un offset de 65536 y se pasa a binario.
4. Se cogen los últimos bits de los siete primeros valores de cada sección.
5. Se pasa al formato ASCII para recuperar el mensaje.

2.3.6. Implementación

Para implementar este sistema de esteganografía se ha creado un programa Matlab 2011a con el que se han obtenido los resultados aquí expuestos. El código de este programa se encuentra en la siguiente sección. La razón de usar Matlab como lenguaje de programación es que como su propio nombre indica (MATrix LABoratory) trabaja muy bien con vectores y matrices, y esto resulta tremendamente útil a la hora de trabajar con imágenes (hipermatrices). Para la implementación del algoritmo en Cloud, se ha usado el programa Octave 3.0 por ser un programa de código abierto, que se instala fácilmente en una instancia de Linux por medio de un script y que, al ser muy parecido a Matlab, pequeñas modificaciones son suficientes para adaptar los códigos a este programa. El código utilizado se muestra en el apartado A.1. del Apéndice A.

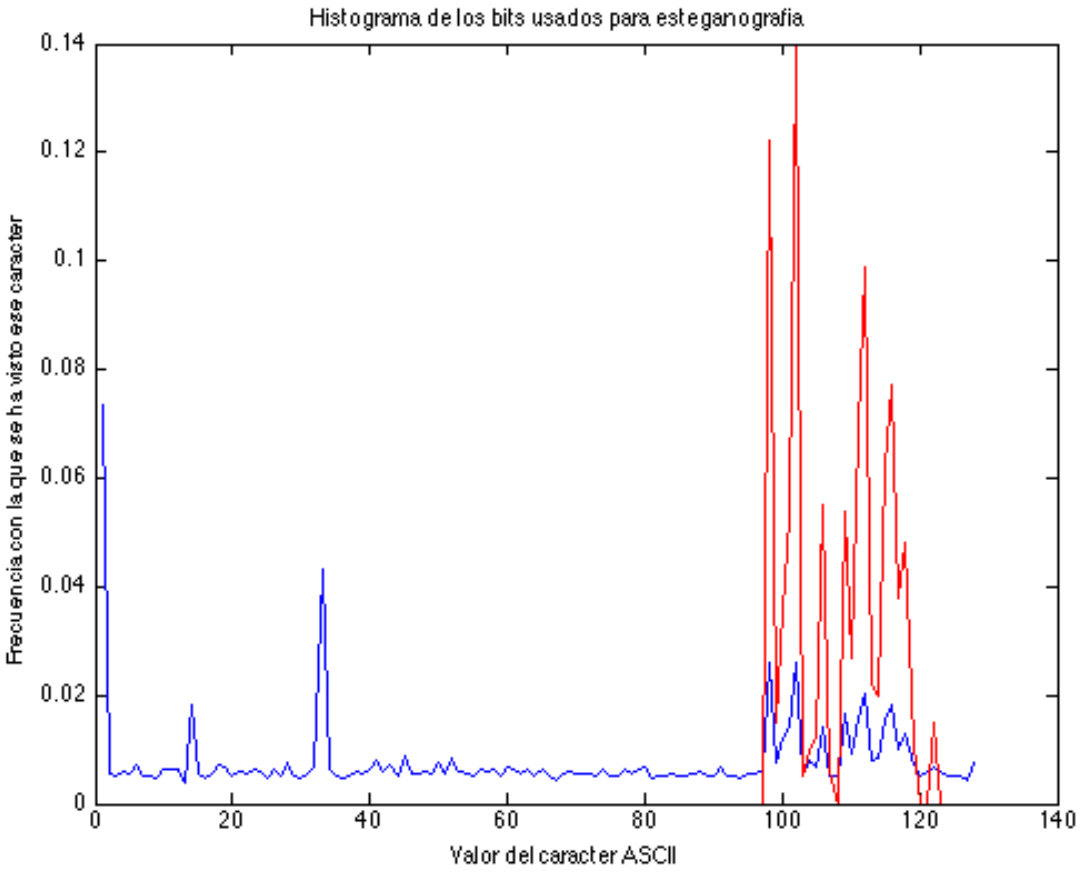

Figura 2.10: *Comparación del histograma de los valores de los píxeles de la imagen original con la esteganografiada de los píxeles utilizados para la esteganografía*

Capítulo 3

Esteganoanálisis

3.1. Introducción

En el capítulo anterior hemos visto una descripción de un modelo genérico de un sistema de esteganografía y la técnica de esteganografía implementada en este trabajo. En este capítulo, debemos olvidarnos del sistema utilizado y tratar de hallarlo comprobando un gran abanico de posibilidades de esteganografía comprobando las estadísticas obtenidas para cada caso, puesto que esto es en definitiva el esteganoanálisis.

Obviamente no seremos capaces de hacer una técnica que contemple todas las posibilidades de esteganografía, pero para el fin de este trabajo, que como hemos explicado en 1.3 es crear una arquitectura de cloud computing que auto-escale de forma inteligente para adaptarse en tiempo de ejecución, nos resultará suficiente explorar un gran conjunto de posibilidades, aunque estás no representen la totalidad.

Es muy importante comprender que la forma en que se realiza el esteganografíado limita o incluso imposibilita la detección del mensaje huésped cuando su longitud es inferior a cierto umbral ya que las estadísticas variarán tan infimamente que resultara indetectable[42].

3.2. Técnica de esteganoanalisis

Como analogía, el estegoanálisis es a la esteganografía lo que el criptoanálisis es a la criptografía. Sin embargo, existe una diferencia fundamental entre un criptoanálisis y un estegoanálisis. Mientras en el criptoanálisis es necesario descifrar el mensaje para considerar que el criptosistema esta roto, en el esteganoanálisis es suficiente detectar la existencia de un mensaje oculto, aunque se desconozca este mensaje para romper el sistema. La razón es que el objetivo del sistema es ocultar la información, no hacerla indescifrable.

Hay una clasificación básica entre los tipos de estegoanálisis:

- **Estegoanálisis manual:** Compara de forma manual las diferencias entre el fichero original y el esteganografiado. Esta técnica tiene inconvenientes muy importantes dado que se necesita tener un fichero original que normalmente no se dispone y aunque se

llegue a detectar que el fichero ha sido modificado, es prácticamente imposible descifrar el mensaje.

- **Estegoanálisis estadístico:** Consiste en el análisis estadístico de los mensajes anfitriones para hallar algún tipo de mensaje oculto.

    1. Estás técnicas conocidas como "blind steganalysis" o esteganoanális a ciegas son muy lentas puesto que clasifican primero el medio, luego las clasifican por técnicas conocidas y finalmente buscan modificaciones sospechosas[42].
    2. Para hallar los mensajes ocultos por algoritmos que usan técnicas de esteganografía habituales se usan técnicas de esteganoanálisis específicas para acelerar y mejorar el proceso, esto es, comparan las estadísticas obtenidas tras las transformaciones que suelen usar los algoritmos de las técnicas habituales para hallar posibles mensajes huésped. Esta segunda técnica es la que se va a emplear para este trabajo. Como inconveniente, la detección de mensajes ocultados manualmente con algoritmos de esteganografía propios serán casi indetectables para estas técnicas de aceleración.

Con objeto de mejorar las técnicas que analizan herramientas comerciales existe una plataforma llamada Steganalysis Detection Architecture (SDA) que tiene una gran documentación de técnicas esteganográficas conocidas y define los elementos que deben constituir una herramienta de estegoanálisis[42].

Las herramientas y algoritmos accesibles para el publico general son normalmente algoritmos que ocultan información en imágenes siendo JPEG el formato más utilizado y por ello las técnicas de esteganoanálisis se han centrado históricamente en imágenes a la hora de analizar posibles mensajes ocultos[28],[25].

Existen numerosas herramientas comerciales, una de ellas es Stegdetect, StegSecret[42]. Estás herramientas ponen de manifiesto que esteganografiar con herramientas comerciales no resulta seguro porque al ser algoritmos de uso publico resulta muy sencillo hallar el mensaje oculto.

Algunos de los algoritmos más utilizados para el esteganoanálisis son aquellos que analizan el bit menos significativo de los píxels de las imágenes entre ellos están el algoritmo chi-square de Andreas Westfeld y Andreas Pfitzmann, el ataque RS-Attack y el ataque PairValues de Jessica Fridrich[42].

3.3. Sistema de esteganoanálisis implementado

En este apartado se presenta el algoritmo de esteganoanálisis que se ha desarrollado para este trabajo. Para ello se presentan primero los casos analizados por el algoritmo de esteganoanálisis creado, a continuación se presentan los resultados obtenidos y finalmente se describe la implementación.

3.3.1. Casos analizados

En este trabajo, se presenta un algoritmo de estegoanálisis estadístico que consiste en analizar directamente varios algoritmos habituales para ocultar mensajes. Para el algoritmo de estegoanálisis que hemos implementado se supone que el idioma usado en el mensaje huésped es el español antiguo o el ingles (Se introduce "El Quijote" como mensaje huésped) en un caso general habría que considerar más idiomas pero para comprobar el correcto funcionamiento del auto-escalado analizar dos resulta suficiente.

Para analizar estadísticamente los idiomas se usa habitualmente la frecuencia de aparición de las letras. En un texto suficientemente largo, la frecuencia de aparición de las letras siempre es muy parecida en un cierto idioma. En el español por ejemplo, la letra "E" tiene una frecuencia de aparición de un 13.68 % siendo la letra más frecuente y la letra "K" con una frecuencia de aparición de un 0.01 % es la letra menos frecuente.

Otra forma de analizar estadísticamente un idioma es la correlación entre letras. Esto significa que por ejemplo, tras la letra "L" lo más probable es que venga una vocal "a,e,i,o,u" y por lo tanto la correlación entre estas dos letras es más alta que entre "L" y otra consonante. Debido a la mayor complejidad de este método, no se ha utilizado para hallar mensajes ocultos, pero podría utilizarse como complemento a las estadísticas anteriores.

En la Figura 3.1 tenemos está frecuencia de aparición de todo el abecedario para el español antiguo y el ingles cuando son codificados en ASCII.

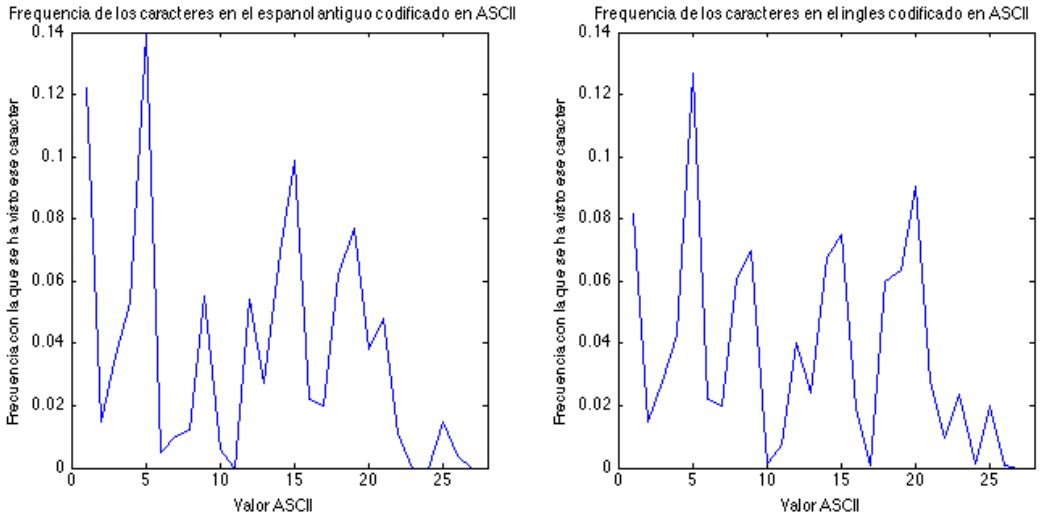

Figura 3.1: *Frecuencia de aparición de las letras en el español antiguo y el ingles codificado en ASCII*

El algoritmo por lo tanto:

1. Crea un bucle que secciona la imagen en todos los tipos de tamaños de bloque posibles.
2. Crea un bucle anidado en el anterior que utiliza diferentes transformadas para el vector creado: FFT, DCT, etc.

3. Por último crea un último bucle anidado que obtiene el valor en ASCII de los bits menos significativos (LSB) de estas transformadas.

4. Finalmente compara los valores ASCII obtenidos con las distintas estadísticas de los lenguajes que se conocen. Esto se hace en dos pasos:

   a) Toma solo los valores que tras codificarse en ASCII toman valores del 97 al 122 que son los correspondientes al abecedario en minúscula y los normaliza para que el total de las frecuencias de aparición de 100 %.

   b) Una vez hecho esto se aplica la función de correlación de los valores obtenidos y cada uno de los idiomas que se quiere analizar. Cuanto mayor sea el mensaje huésped, el valor de la correlación más se acercará a 1.

Obviamente este algoritmo no analiza todas las posibilidades existentes para analizar si una imagen a sido o no sometida a un proceso de esteganografía: se podría codificar un bit que no es el último, hacerlo siguiendo un patrón por el método pseudorandom, etc. o usar otras técnicas que hemos explicado anteriormente como sustitución o distorsión. Sin embargo, para el análisis posterior que lo que trata es de optimizar un algoritmo de esteganoanálisis usando cloud computing, este algoritmo resulta suficiente.

3.3.2. Resultados del Esteganoanálisis

Cuando en este bucle anidado se llega al tamaño de bloque, transformada correcta, y lenguaje correcto se llega a que los bits que se analizan tienen una alta correlación con las frecuencias de los caracteres del idioma en cuestión. En el bucle podemos establecer una condición como por ejemplo que la correlación sea $corr > 0{,}8$ de forma que cuando lleguemos a alguna configuración con una correlación mayor nos muestre los caracteres ASCII para ver si existe un mensaje oculto. En la Figura 3.2 se muestra resaltado en negrita la iteración en la que se halla el mensaje.

La Tabla 3.1 muestra las correlaciones obtenidas al iterar con el tamaño de la sección y la iteración en la que se obtiene el resultado resaltado en negrita.

El algoritmo de esteganoanálisis analiza muchos más tamaños de sección, sin embargo, esta tabla sirve para mostrar como se detecta por medio de la correlación donde está el mensaje oculto (en el capítulo anterior se explicaba que el tamaño de sección usado era de 8x8, por lo tanto el algoritmo de esteganoanálisis funciona). El algoritmo itera además con varias transformadas y dos lenguajes que producen tablas similares.

3.3.3. Tiempo necesario para obtener el resultado

Esta sección sirve como muestra de la utilidad de la computación distribuida para el esteganoanálisis

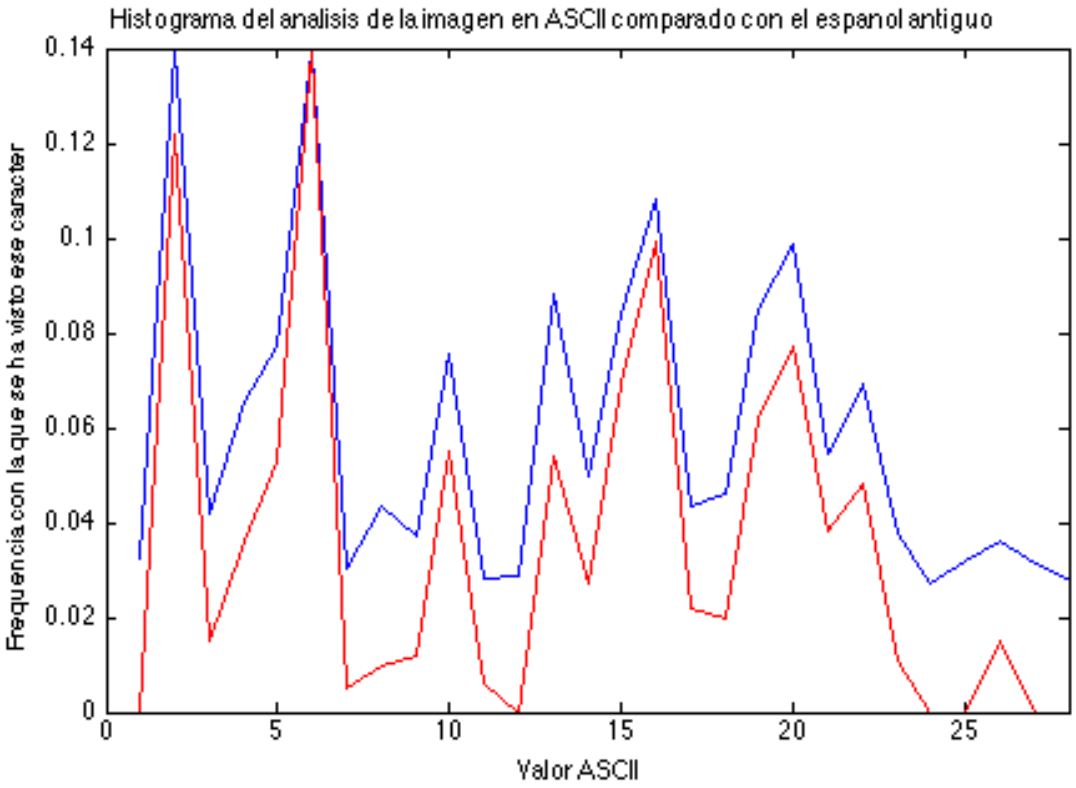

Figura 3.2: *En este momento, se detecta que hay una probabilidad muy alta de que haya un mensaje oculto. En concreto, esta es la configuración que hemos usado para la esteganografía y la correlación es 0,9878 por lo tanto muy próximo a la probabilidad del lenguaje español antiguo.*

Obsérvese en primer lugar que el algoritmo de esteganoanális usado es un proceso totalmente paralelizable puesto que se trata de un bucle anidado sin dependencias entre iteraciones. Por lo tanto, podemos enviar la misma imagen a muchos procesadores y que cada uno de ellos analice una posible alternativa de esteganografía.

La Tabla 3.2 se obtiene considerando el caso anterior de analizar todas las secciones desde 5x5 hasta 10x10. Se puede observar que en un tiempo de 19.3615 segundos más el tiempo de transmisión de la imagen podría analizarse completamente la imagen usando 36 procesadores como el usado en este trabajo asignando a cada uno un tamaño de sección.

Sin embargo, esto no sería lo más eficiente porque el procesador que analizara usando secciones de 10x10 estaría inactivo durante casi 13 segundos. Por lo tanto, es necesario un algoritmo que elija cuantos procesadores son necesarios y como elegir que trabajo mandar a cada uno para que el proceso completo no tarde más de los 19.3615 segundos usando menos de 36 procesadores.

Adaptar está técnica para cloud computing será el siguiente puntode este trabajo.

Tamaño de sección						
Vertical / Horizontal	5	6	7	8	9	10
5	0.071	-0.311	0.191	-0.185	0.056	0.225
6	0.1	-0.005	0.089	-0.202	0.07	0.099
7	-0.021	-0.014	0.044	0.179	0.134	0.036
8	-0.006	0.14	0.027	**0.988**	0.537	-0.034
9	0.009	0.106	0.035	-0.104	-0.099	0.268
10	-0.006	-0.269	0.129	-0.182	0.151	-0.263

Cuadro 3.1: *Valores de la correlación del bit menos significativo para distintos tamaños de sección*

Tamaño de sección						
Vertical / Horizontal	5	6	7	8	9	10
5	**19.36**	16.42	13.82	11.89	11.22	10.74
6	16.72	14.29	11.96	10.27	10.19	8.95
7	15.28	12.72	10.97	9.29	8.88	8.38
8	14	11.63	10.16	8.4	8.13	7.63
9	13.03	10.7	9.45	7.91	7.49	7.17
10	11.87	10.08	8.67	7.52	6.95	**6.71**

Cuadro 3.2: *Tiempo en segundos en realizar el esteganoanálisis para distintos tamaños de sección.*

3.3.4. Implementación

Para implementar este sistema de esteganografía se ha creado un programa en Matlab 2011a con el que se han obtenido los resultados aquí expuestos y cuyo código se muestra en el apartado A.2. del Apéndice A.

Capítulo 4

Cloud Computing Esteganoanálisis

4.1. Introducción

En el capítulo anterior hemos visto el algoritmo que se usa para el esteganoanálisis. Como hemos visto el algoritmo contiene bucles anidados independientes que usan la misma imagen como dato de entrada y producen una salida: la correlación entre las estadísticas obtenidas y las estadísticas de los idiomas.

Tener bucles anidados independientes lo convierte en un algoritmo altamente paralelizable que puede beneficiarse de las características de la computación distribuida para acelerar el análisis.

En este capítulo se va a adaptar el algoritmo de las secciones anteriores a un entorno cloud computing con arquitectura master-slave auto-escalable. La arquitectura master-slave sigue un modelo de programación centralizado donde un proceso lleva el control y el resto se encargan del trabajo de cómputo. Por otro lado, se define una arquitectura de cloud computing auto-escalable como aquella que por medios automáticos: bien sea con disparadores que reaccionan a ciertas condiciones como carga de CPU, RAM, etc. o activos haciendo predicciones de carga, arrancan o apagan instancias escalando los recursos disponibles. Para probar la nueva arquitectura se usara el proveedor Amazon EC2 por tratarse del proveedor de servicios de cloud computing líder del mercado.

4.2. Introducción al Cloud Computing

Cloud computing es el nombre que se le ha dado a un nuevo paradigma de la computación. La idea principal del cloud computing es revolucionaria, y supone ofrecer la capacidad de computo como servicio y "prescindir" de la idea de capacidad instalada que teníamos hasta hace poco[27]. Tambien es muy importante destacar que puede reducir drásticamente los tiempo de ejecución de aplicaciones paralelizables porque 1 máquina en funcionamiento durante 1000 horas cuesta lo mismo que 1000 máquinas funcionando durante 1 hora. A pesar de ser una idea revolucionaria es una tecnología que se encuentra aún en su infancia y existen muchos problemas por resolver[48].

4.2.1. Clasificación de sistemas cloud computing

Los sistemas de cloud computing se pueden clasificar siguiendo varios criterios. Los criterios más comunes son en función del tipo de servicio ofrecido y en función de su localización.

Clasificación en función del tipo de servicio ofrecido

Existen varios tipos de cloud computing en función del tipo de servicio que ofrecen[26]:

- **Infrastructura como Servicio (IaaS):** El proveedor ofrece una máquina virtual al cliente en la que pueda gestionar todo como si se tratara de una máquina local. Gracias a esto, el usuario consigue desplazar una serie de problemas relacionados con la gestión de las máquinas al proveedor sin perder la capacidad de modificar las características del servicio y el entorno. Algunos ejemplos de IaaS son: Amazon EC2 S3[3], Go Grid[12], Cloud Sigma[8], etc.

- **Plataforma como Servicio (PaaS):** El proveedor resuelve muchos problemas que plantean al cliente las IaaS como instalar, configurar y mantener sistemas operativos, sistemas de bases de datos, servidores de aplicaciones, etc. Estos vienen por defecto en la PaaS y el usuario, solo se tiene que preocupar de las aplicaciones que quiere instalar. Pero las ventaja que esto supone con respecto a IaaS también tienen su contrapartida. PaaS tiene limitaciones en el entorno de ejecución, precisamente, porque todo está definido por defecto. Algunos ejemplos de este tipo de servicio son: Google App Engine[13], Azure de Microsoft[6], etc.

- **Software como servicio (SaaS):** El proveedor hace de administrador y host de las aplicaciones a la vez. Ejemplos de este tipo de servicio son: Oracle CRM On Demand[16], Force de Salesforce[17], Net suite[14] , etc.

Clasificación en función de la localización

En función de la localización de los recursos los clouds se diferencian en cloud privados o Internal Clouds, cloud públicos o External Clouds y clouds híbridos[26]. Además de la localización esta clasificación también afecta a la seguridad, coste, disponibilidad y otros factores[34].

- **Clouds Privados:** Aplican los conceptos de cloud computing a recursos propios de la empresa. Lo que se hace es coger los recursos de toda la empresa en vez de por departamentos como se hacía anteriormente siendo así más eficientes y teniendo menor coste que los sistemas anteriores.

- **Clouds Públicos:** Son los clouds en los que algunos recursos o servicios son prestados por una compañía externa a través de navegadores web o interfaces de programación de interfaces (Aplication Programming Interface) APIs. Son, desde el punto de vista de la arquitectura, iguales que los cloud privados pero a una mayor escala. Gracias

a la diversificación de localizaciones y industrias consiguen estar casi exentos de la variabilidad horaria y otros factores de variabilidad lo que les permite ser más eficientes en el uso de recursos.

- **Clouds Híbridos:** Los clouds híbridos son una mezcla de los dos anteriores. La idea es tener un cloud privado que se usa para aplicaciones cuya seguridad pueda ser más comprometida o para los requisitos de computación estables para el que un cloud privado puede ser más baratoy un cloud público para el resto de aplicaciones.

4.2.2. Tipos de instancias

Una instancia es una cantidad de capacidad de computo dedicada que se ofrece al usuario mediante tecnologías de virtualización. Al igual que el esquema de precios, es importante ver los tipos de instancia que existen en la actualidad en los proveedores de cloud computing y cual de ellos es el más adecuado para nuestro propósito[41].

Los tipos de instancias pueden estar preconfigurados o se pueden construir a medida dependiendo del proveedor cloud. En el caso de Amazon EC2 que es el proveedor cloud que se considera para el proyecto, las instancias disponibles están preconfiguradas y son las siguientes[41], [4]:

- **Instancias Estándar:** Son instancias que se comportan bien con la mayoría de las aplicaciones. Amazon EC2 tiene dos tipos de instancias estándar: Las pequeñas que son las que se usan por defecto y las grandes.
- **Instancias Micro:** Son instancias muy pequeñas que se comportan bien cuando la aplicación no requiere demasiado computo pero tiene que estar siempre funcionando, un ejemplo típico de aplicación que se adapta bien a esta instancia son las páginas web con poco tráfico.
- **Instancias con mucha memoria:** Son instancias con una memoria sobre-dimensionada con respecto a su CPU que se comportan muy bien con aplicaciones de gran throughput como aplicaciones de bases de datos.
- **Instancias con mucha CPU:** Estás instancias tienen un CPU sobre-dimensionado con respecto a su memoria y son apropiadas para aplicaciones que requieren gran cantidad de cómputo.
- **Instancias de tipo Cluster:** Estás instancias tienen una CPU proporcionalmente grande y unas conexiones de red mejoradas. Son apropiadas para aplicaciones con mucho tráfico de red y de altas prestaciones.
- **Instancias de tipo Cluster con GPU:** Son instancias especialmente diseñadas para beneficiarse de la alta paralelización. Son óptimas para aplicaciones de rendering y con medios audio-visuales

4.2.3. Esquema de precios

En el mercado actual de cloud computing hay tres esquemas de precios dominantes: El pago on-demand donde el usuario paga por los recursos utilizados; el pago con reserva donde el usuario paga por adelantado la reserva de ciertos recursos y después obtiene un mejor precio por su uso; y finalmente el sistema de pago spot en el que el usuario determina el precio máximo por instancia y hora que está dispuesto a pagar y el proveedor cloud le sirve el recurso solo bajo ciertas condiciones de demanda[21].

Es el esquema de precios del cloud computing lo que ha producido una revolución en el sector, porque es lo que ha hecho esta tecnología competitiva[23]. Existen problemas tecnológicos que hay que superar aún, pero si el esquema de precios sigue siendo competitivo los problemas tecnológicos se solventaran y el cloud computing dominará el mercado. Es por esto, por lo que conviene ver en detalle los esquemas de precios[30].

- **On-demand:** Es el esquema de precio dominante en el mercado actual de cloud computing y su idea principal es pagar solo por lo que se usa. Esto ha supuesto una revolución porque supone olvidarse de la capacidad instalada en los servicios de computación. Lo ofrecen la mayoría de los proveedores cloud: Amazon[5], GoGrid[11], CloudSigma[7], Elastic Host[9], etc. A pesar de estar englobados en el esquema de precios on-demand existen diferencias entre proveedores: Algunos como Amazon[4] ofrecen instancias preconfiguradas con ciertas características como la estándar con 1.7GB de RAM, 1 Core con 1GHz y 160GB de capacidad por 0.085$/h. Otros como Elastic Host[9] no ofrecen instancias preconfiguradas sino que se paga por cada GB de Memoria, Ghz de procesador, etc. cierta cantidad.

- **Reserva:** Otro esquema de precios muy importante en el mercado actual es el pago con reserva. La idea es pagar por adelantado una cierta cantidad para reservar un número de instancias, que se amortizan con un precio inferior al on-demand durante su uso. Al igual que el esquema de precios on-demand es ofrecido por la mayoría de los proveedores cloud: Amazon[5], CloudSigma[7], etc. Y como ocurre en las políticas de precios on-demand también existen ciertas diferencias entre las políticas de precios con reserva. En la mayoría de los proveedores cloud se requiere un pago único por adelantado para reservar las instancias durante cierto tiempo y en ese periodo de tiempo, el precio por usar las instancias es menor. Este es el caso de Amazon EC2 que para el caso de una instancia estándar tiene el precio para la reserva de 1 año en 227.5$ con un coste posterior de 0.04$/h. Existe sin embargo una variante que se aplica por ejemplo en GoGrid[11] donde se reservan un número determinado de recursos y no hay que pagar nada más hasta agotar los recursos reservados.

- **Spot:** El último esquema de precios que está ampliamente extendido es la configuración de precios spot. La idea principal es que el usuario establezca un techo de gasto para una instancia y que, en función de la carga de los servidores del cloud provider, este le ofrezca o no el servicio. Esta configuración suele ser la más barata pero no ofrece garantías de servicio. Si el precio definido por el usuario es 0.04$/h y el precio

al que ofrece el servicio el proveedor supera en cierto momento ese umbral, el servicio se interrumpe y la instancia se apaga.

4.2.4. Ventajas del cloud computing respecto a otros sistemas de computación distribuida

Como hemos visto en el capítulo anterior y hemos enfatizado al principio de este capítulo, en este proyecto vamos a aplicar cloud computing para resolver el problema del esteganoanálisis de imágenes. Hemos visto que en general, un sistema de computación distribuida mejora enormemente el tiempo de computación de nuestro algoritmo de esteganoanálisis por ser altamente paralelizable. En esta sección, analizaremos primero las diferencias entre cloud computing y otros sistemas de computación distribuida. Tras este análisis, veremos que ventajas nos ofrece cloud computing con respecto a otros sistemas de computación distribuida como pueden ser un grid, un cluster de computación[33].

Diferencias entre cloud computing y otros sistemas de computación distribuida

Cloud computing es un concepto de computación distribuida que ha evolucionado del grid computing. Por ello, mantienen muchas características comunes a pesar de ser dos tecnologías distintas. La idea del grid computing cuando se creo era ofrecer al usuario capacidad de computo bajo demanda, al estilo de lo que ocurre con la electricidad en la red eléctrica. La visión por tanto, es la misma que en cloud computing: ofrecer capacidad de computo a bajo coste, fiable y flexible mediante servidores operados por un tercero.

Sin embargo, hay una serie de razones que han hecho evolucionar al grid computing hacia cloud computing y han diferenciado las dos tecnologías[33].

1. Los costes de virtualización han bajado, y los microchips actuales suelen incluir hardware para soportar virtualización, reduciendo la penalización de rendimiento que suponía virtualizar los recursos hasta hace poco[46].

2. Los proveedores cloud han invertido billones de dolares en construir grandes sistemas.

Esto ha creado una serie de diferencias entre grid computing y cloud computing:

1. Cloud computing encapsula una serie de servicios por medio de virtualización que puede ofrecer a usuarios fuera del cloud, lo que permite ofrecer una serie de recursos y servicios más abstractos.

2. Cloud computing se guía por economías de escala porque los proveedores cloud compran una gran cantidad de equipamiento lo que les permite obtener un precio más barato[34].

3. Gracias a la virtualización el servicio ofrecido por cloud computing se puede configurar el servicio dinámicamente.

Todas estas características están ausentes en grid computing. Hemos visto una serie de características del cloud computing que no tiene el grid. Veamos ahora las características del grid. Ian Foster[32] definió las características que debe tener un grid.

1. El grid debe ser un conjunto de recursos que se coordinan de manera descentralizada.
2. Las interfaces de control del grid deben ser estándar y de propósito general.
3. Los grids deben ofrecer características de calidad de servicio no triviales.

En esta definición vemos como el cloud computing no cumple las tres condiciones. El cloud computing suele estar coordinado por el proveedor del servicio y por lo tanto, tiene el control centralizado. Además existen diferentes APIs para comunicarse con el cloud y estas no son estándar[33]. La tercera condición si se cumple puesto que también trata de ofrecer una calidad de servicio establecida.

Es importante saber, que aunque cloud computing ha evolucionado de grid computing, este sigue teniendo su mercado, especialmente en el ámbito de las universidades. Una de sus mayores ventajas frente al cloud es la interfaz estándar. La Figura 4.1[33] nos muestra muy bien el mercado de cloud computing, web 2.0, clusters, supercomputadores y grids en función de si son orientados a aplicación o a servicios y su escala.

El grid se orienta hoy en día a compartir recursos y coordinar la resolución de problemas a través de diferentes organizaciones especialmente entre universidades. El estándar de facto para ello es Globus toolkit[10].

Ventajas de cloud computing

La virtualización de los recursos es una de las mayores ventajas del cloud computing. La virtualización proporciona una abstracción respecto a la infraestructura sobre el que se sustenta y facilita crear un pool de recursos que facilitan consolidar la carga mejorando la eficiencia. Además, la virtualización también posibilita encapsular las aplicaciones con su configuración mejorando la seguridad, manejabilidad e isolación de las aplicaciones. El uso de la virtualización también mejora la disponibilidad gracias a la recuperación rápida[33].

Cuando se habla de virtualización hay que tener presente que actualmente la perdida de rendimiento asociada a la virtualización ya no es tan destacada como antes porque los fabricantes de procesadores han incluido hardware dedicado para soportar la virtualización[46].

Los grids no se sustentan tanto en la virtualización lo que no les permite obtener estas ventajas.

Una ventaja muy importante de las aplicaciones cloud es su alta disponibilidad. Históricamente, la disponibilidad de las aplicaciones se ha conseguido por medio de los nodos replicados. De esta forma, se conseguía que ante el eventual fallo de uno de los nodos el otro prosiguiera con la aplicación. Sin embargo, este método no ofrecía protección contra fallos catastróficos. Un terremoto, una inundación, etc. podían dejar a todos los nodos replicados inutilizables interrumpiendo la aplicación.

Cloud computing ofrece una solución a este problema con las zonas de disponibilidad o availability zones[24]. Las zonas de disponibilidad son servidores que se encuentran en lugares

físicos separados y con todos los sistemas replicados. Si lanzamos un servidor A a la zona 1 y un servidor replicado B a la zona 2 es extremadamente difícil que se produzca un fallo simultaneo incluso en el caso de una catástrofe. Por esto, las aplicaciones que necesitan una alta disponibilidad se pueden beneficiar de las ventajas del cloud computing.

Otra ventaja del cloud computing con respecto a otros sistemas de computación paralela es su versatilidad. La versatilidad se obtiene "prescindiendo" de la idea de una capacidad instalada adaptándose así a la demanda dinamicamente.

Veamos esto con un sencillo ejemplo:

Imaginemos que una empresa quiere abrir una página web en breve. Hasta hace poco, lo que tenía que hacer la empresa era comprar una serie de servidores para que pudieran responder a las peticiones de los usuarios. Esto suponía un problema porque implicaba una gran inversión inicial que, en el caso de que la página web no cumpliera las expectativas, caía en desuso y, una demanda superior a la esperada, congestionaba los recursos. Con cloud computing esto cambia, ya que gracias al pago on-demand de los recursos, ya no resulta necesaria una gran inversión inicial y el número de servidores que atienden las peticiones puede cambiar con el tiempo. De este modo, se reduce la inversión inicial y con ella el riesgo de abrir una página web. Esto se conoce como transferencia de riesgo (transference of risk) porque con cloud computing la inversión inicial la hace el proveedor de cloud computing, no la empresa que quiere abrir una página web[22].

4.2.5. Auto-Escalado de servicios de Cloud Computing

Escalar una aplicación en cloud computing significa aumentar o disminuir los recursos de computación disponibles arrancando o apagando instancias.

La forma de afrontar la escalabilidad de las aplicaciones tradicionalmente, ha sido anticiparse a la carga máxima y comprar la infraestructura necesaria para soportar esa carga[24].

Sin embargo, esto no se puede aplicar o presenta dificultades para ciertas aplicaciones porque la carga máxima debe fijarse por adelantado y la infraestructura instalada es difícil de cambiar. Además, la infraestructura no operara a pleno rendimiento salvo en el caso de pico de demanda, lo que produce una infrautilización de los recursos.

El escenario óptimo para una aplicación sería aumentar o disminuir la infraestructura en función de la carga de trabajo actual. Esto, es posible gracias a las tecnologías de virtualización y el pago on-demand presentes en cloud computing.

En cloud computing la la cantidad de recursos del proveedor es virtualmente ilimitada y el pago on-demand junto a la virtualización de las máquinas permite encender o apagar máquinas sin necesidad de tener una capacidad instalada.

Todo esto, conlleva un coste muy pequeño en comparación con el sobre-dimensionamiento de la infraestructura que se llevaba a cabo antiguamente.

Escalar en cloud computing es fácil , pero el auto-escalado o escalado de forma automática presenta una serie de problemas que veremos a continuación. Pero antes, vamos a ver las técnicas de auto-escalado disponibles en la actualidad.Existen dos tipos fundamentales de auto-escalado en función del momento en que se toma la decisión de escalar:

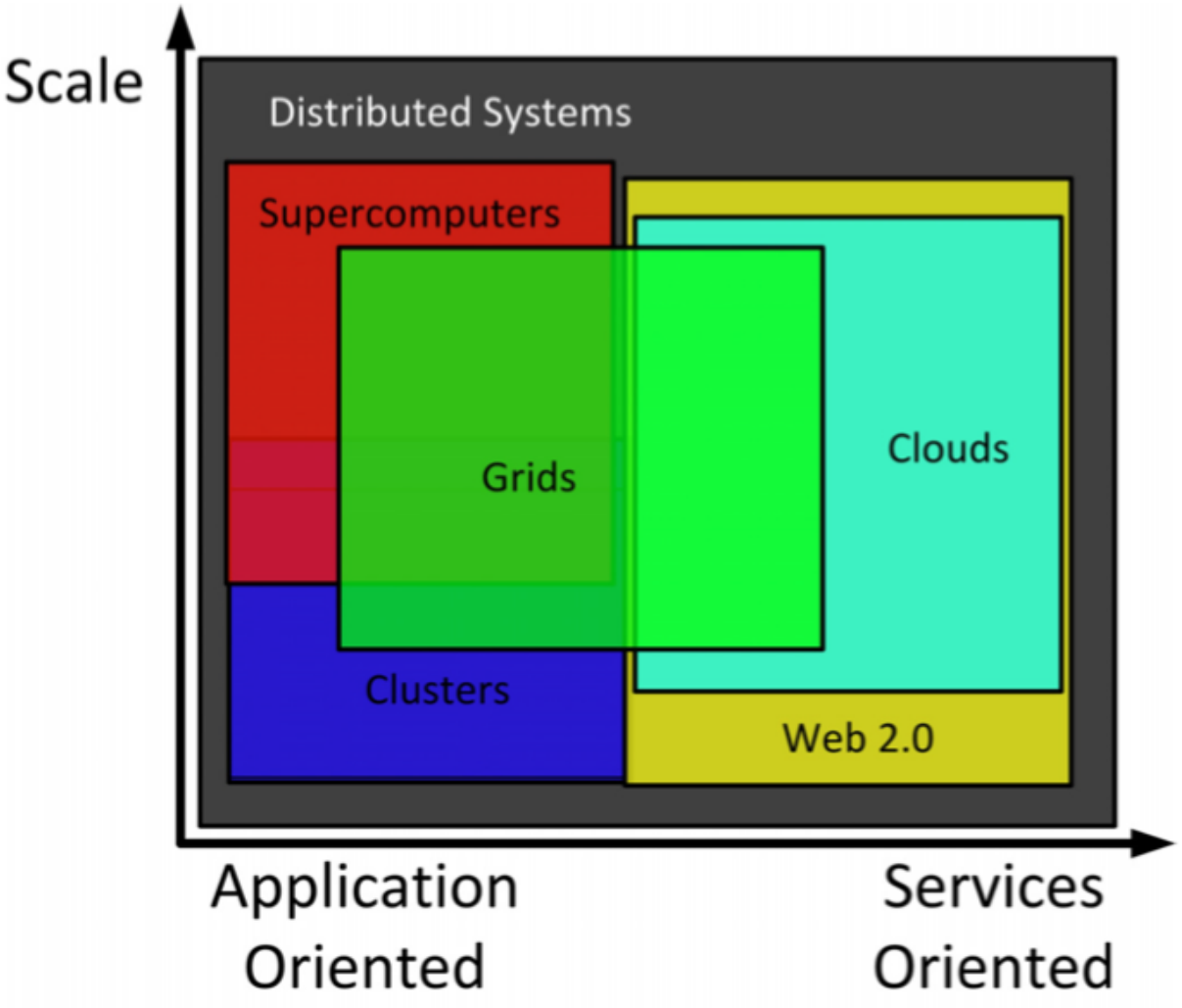

Figura 4.1: *Comparación entre tecnologías Grid, Cluster y Cloud*

1. **Auto-Escalado reactivo:** Es el más usual de las técnicas de auto-escalado. La idea principal de estas técnicas es tomar decisiones de escalado en función del estado actual de los servidores. El administrador, fija una serie de condiciones que pueden ser simples; como máximo y mínimo de CPU, RAM, etc; o complejas, como algoritmos que usan parámetros actuales como valores de entrada y producen una salida que se usa para tomar las decisiones de auto-escalado.

2. **Auto-Escalado activo:** Debido a que arrancar nuevas instancias y configurarlas lleva un tiempo considerable de 5-10 minutos. Las técnicas de auto-escalado activo, tratan de adelantarse a las necesidades modelando matemáticamente el uso de los servidores en el pasado y usando estos modelos para realizar predicciones de uso de servidores en el futuro.

Otra clasificación de las técnicas de escalado se hace en función de la forma en la que escalan el servicio[47].

1. **Escalado horizontal:** Es la técnica de escalado más habitual. Consiste en añadir una nueva instancia a una serie de instancias ya arrancadas y luego utilizar un balanceador de carga para distribuir la carga total entre el nuevo número de instancias.

2. **Escalado vertical:** Esta técnica es más complicada. La idea principal es asignar más recursos a una instancia ya arrancada. El problema es que la mayoría de los sistemas operativos actuales no soportan el cambio de los recursos asignados de forma dinámica. Esto ha llevado a diversos investigadores a estudiar formas similares de escalar las aplicaciones. Una de las ideas es apagar una instancia con ciertos recursos y arrancar otra instancia replicada, pero con más recursos[47].

Todas estás técnicas de auto-escalado presentan los siguientes problemas en la actualidad:

- Los proveedores cloud ofrecen una auto-escalabilidad por medio de disparadores o trigers que se definen a partir de métricas de rendimiento. Estos disparadores lanzan o apagan un número definido de instancias por lo que no tienen en cuenta el efecto no-lineal que se produce, porque no es lo mismo pasar de 1 a 2 instancias que de 100 a 101. Además, las métricas utilizadas: uso de CPU, disco, etc. son métricas de nivel de infraestructura no de Calidad de Servicio (Quality of Service QoS) y esto también puede suponer un problema para ciertas aplicaciones en las que las métricas de nivel de infraestructura no están directamente relacionadas con el QoS[40],[1].

- El tiempo de inicio de las instancias no se considera en las técnicas reactivas. Pero incluso en las técnicas activas, no se considera la variabilidad existente en el tiempo de arranque.

- Tampoco se considera el tiempo de apagado de las instancias, que aunque más breve también es considerable en algunos casos.

- Otro gran problema es el pago por horas, que los disparadores no consideran. Esto es muy importante porque a pesar de tener una instancia ociosa, no nos interesa apagarla en el minuto 40 de facturación, porque ya hemos pagado la hora completa.

- Las razones expuestas en los puntos anteriores obligan a definir en muchos casos la inteligencia o algoritmo de auto-escalado en la misma aplicación[38] . Se llama inteligencia del auto-escalado a la toma de decisiones de los algoritmos que deciden cuando aumentar o disminuir los recursos reservados siguiendo una serie de reglas.

- Definir la inteligencia de auto-escalado en la misma aplicación supone un problema adicional y es que los proveedores cloud no ofrecen herramientas de escalabilidad homogéneas lo que implica que portar una aplicación que deba auto-escalar de un proveedor a otro supone reescribir gran parte del código[38].

- La escalabilidad de la red no se suele considerar habitualmente[47]. Esto puede suponer un problema porque el número creciente de instancias puede incurrir en un incremento de ancho de banda requerido que potencialmente llegue a saturar la red.

Ante estos problemas existen numerosos trabajos de investigación que presentan las siguiente soluciones:

- Usar métricas de nivel de aplicación y un algoritmo de auto-escalado interno[40].
- Hacer predicciones de las necesidades que se van a tener y guarda un registro con el tiempo variable de inicio de las instancias para anticiparse al tiempo de arranque variable de 5-10 minutos.
- Capturar la inteligencia de la auto-escalabilidad en una parte del código que es ajena a los cambios entre los diferentes proveedores cloud, evitando así que se deban conocer muchos mecanismos de escalabilidad diferentes[38].

4.3. Análisis del algoritmo para decidir que arquitectura utilizar

Para elegir la arquitectura que vamos a utilizar, en primer lugar identificaremos las variables que influyen en el rendimiento del algoritmo y analizaremos cada una de ellas. Tras el análisis detallado de cada una de ellas tomaremos una decisión sobre la arquitectura que se adapte lo mejor posible a este algoritmo.

4.3.1. Variables que influyen en el rendimiento del algoritmo

Para optimizar el rendimiento de este algoritmo tenemos que analizar varias variables: Paralelismo del algoritmo, cantidad de memoria y CPU utilizado y cantidad de transmisión de datos necesario para cada iteración.

En función de este análisis debemos decidir el tipo y número de instancias que hay que usar en el sistema de cloud computing y su arquitectura.

- En función del paralelismo del algoritmo fijamos un límite superior de instancias para resolver el problema.
- En función del uso de memoria y CPU elegiremos de entre las instancias ofrecidas por Amazon EC2 si utilizar instancias high cpu, high memory o standard. En la sección 4.2.2 analizamos las características de estas instancias y en la sección 4.3.1 explicaremos en qué se basa la decisión de usar instancias high CPU basándonos en las características del algoritmo.

- En función de la cantidad de datos transmitida en cada iteración podemos decidir si nos interesan muchas instancias pequeñas capaces de correr pocos threads simultaneamente pero más baratas o solicitar instancias más grandes capaces de correr más threads y por lo tanto, que generen menos tráfico en la red al necesitar una sola copia de la imagen. En la sección 4.3.1 concluimos que las instancias grandes son las más idóneas para este proyecto. También podemos ver como las arquitecturas influyen en la cantidad de datos transmitidos y elegir en consecuencia entre distintas arquitecturas.

Número máximo de instancias para procesar cada imagen

Como hemos explicado en el capítulo anterior, el algoritmo de esteganoanálisis usa un bucle que recorre todas las opciones que hay para ocultar un mensaje. En adelante, supondremos que cada opción o iteración del bucle será procesado por una sola instancia como vemos en la Figura 4.2 aunque en un caso general, también se podría paralelizar cada iteración.

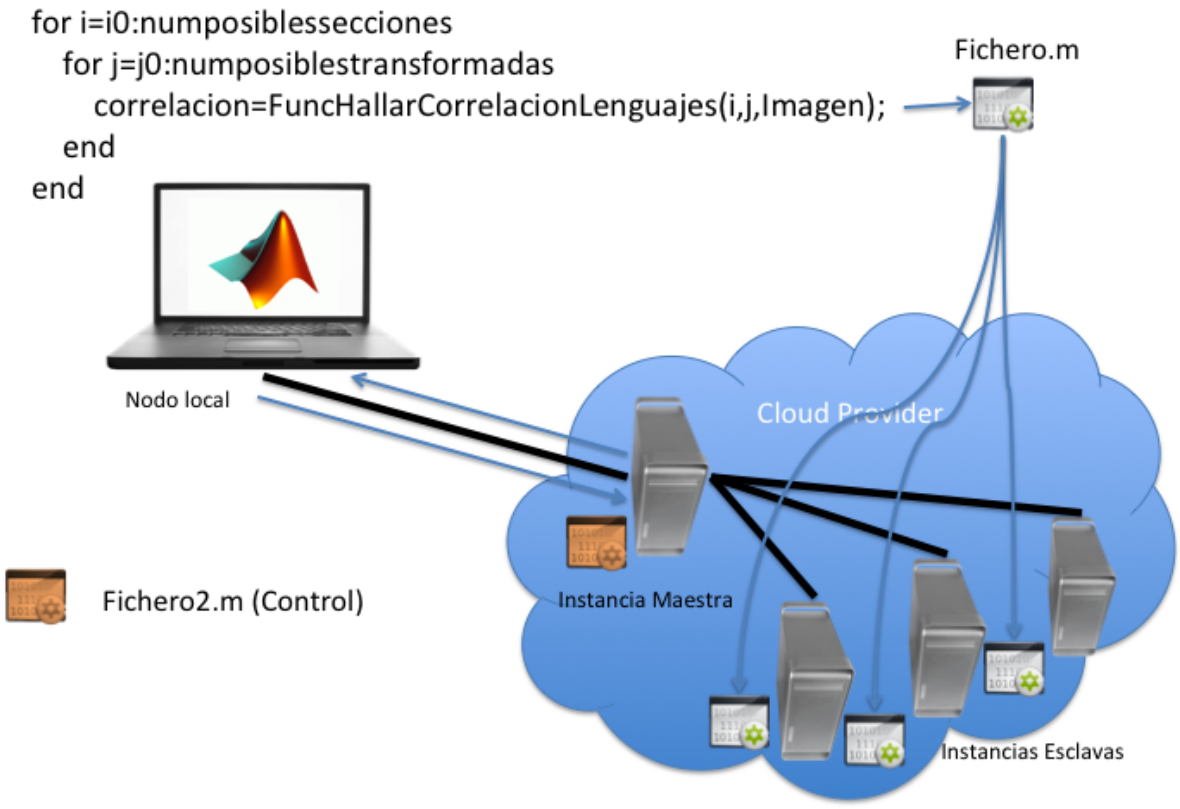

Figura 4.2: *Paralelización del bucle*

Con la condición de no dividir el análisis de una iteración del bucle en varias instancias,

podemos calcular cuanto tardaríamos en analizar una imagen si paralelizaramos completamente el bucle. El tiempo de análisis de la imagen será el mismo que el tiempo de análisis de la iteración más lenta dentro del bucle anidado de nuestro algoritmo.

Si paralelizaramos totalmente el algoritmo y asignamos a cada iteración del bucle anidado un thread o hilo de ejecución tenemos:

$$num_{threads} = num_{secciones} \times num_{transformadas} \tag{4.1}$$

No es necesario un bucle anidado para los lenguajes porque calcular la correlación de las estadísticas obtenidas con las estadísticas del lenguaje no requiere mucho cómputo y sí los pasos anteriores por lo que paralelizarlo es contraproducente.

Para instancias con varios cores podemos enviar un ejecutable y ejecutarlo tantas veces como procesadores tenga disponibles con distintos parámetros. Por tanto, el número de instancias totales será:

$$num_{instances} = num_{threads}/num_{coresperinstance} \tag{4.2}$$

Llegar a este paralelismo es ineficiente porque el tiempo en analizar cada iteración varía y tendríamos instancias inactivas durante largos periodos de tiempo. Sin embargo, de este análisis obtenemos el número máximo de instancias que necesitamos para procesar las imágenes a analizar en el periodo de tiempo deseado.

Si el algoritmo de auto-escalado determina que son necesarias más instancias no se terminará el trabajo a tiempo porque el auto-escalado no producirá ninguna aceleración. La razón es que, al llegar al límite de paralelización del algoritmo, no obtendremos ninguna ventaja añadiendo más instancias. En la sección 4.3.3 se obtendrá con más precisión el número de instancias necesarias.

Análisis teórico del tráfico de datos producido por el algoritmo

Cada instancia requiere del siguiente tráfico de datos para funcionar:

1. **Programa ejecutable:** En el caso de crear una arquitectura con estructura de árbol master-slave, tendremos que pasar al nodo principal dos tipos de ficheros: Uno para los nodos que tienen más instancias a su cargo (master), y otro tipo de ejecutable para instancias esclavas. Estos fichero se crean modificando el código del apartado A.2. del Apéndice A desenredando el bucle anidado y modificandolo para que sirva para Octave 3.0[15]. Octave 3.0 es un programa de código abierto compatible en gran medida con Matlab 2011a. Para ejecutar código Matlab en Octave solo hace falta modificar pequeñas partes de código que usan funciones especiales de Matlab. La decisión de usar Octave 3.0 en vez de Matlab 2011a en Amazon EC2 es que Octave es fácilmente instalable mediante scripts y al ser de código abierto funciona en instancias Linux cuyo coste en Amazon es inferior a las instancias Windows[5]. El .m enviado a los nodos ocupa 7.5KB.

2. **Datos del programa:** Cada instancia necesita recibir una vez la imagen sobre la que realizar el esteganoanálisis. Esta imagen se transmitirá por medio de un fichero del tipo .mat que se usa en Matlab para guardar matrices. Además de los datos a analizar es necesario que el master o nodo principal del árbol envíe a cada instancia la sección y transformada a analizar para que cada instancia analice una iteración distinta. Esto se hace a través de caracteres de entrada.

3. **Resultados del programa 1:** Las instancias slave devolverán la sección y transformada analizado y la correlación existente para cada idioma. Devolver la información sobre la sección y la transformada es necesario porque una instancia puede analizar más de una iteración y en el caso de no especificar la iteración analizada el nodo principal no sabe a que iteración corresponde el resultado.

4. **Resultados del programa 2:** En el caso de que la correlación obtenida supere cierto umbral (e.g. corr>0.8) se devolverá también el mensaje obtenido. Observase que cuanto mayor sea el umbral será más probable que el mensaje obtenido tenga un mensaje oculto pero, sin embargo, los mensajes cortos pueden quedar ocultos por no superar el umbral de correlación al no alterar las estadísticas lo suficiente.

5. **Datos de rendimiento:** Las instancias también deben enviar datos acerca de su rendimiento: CPU (Cores) utilizados, memoria, etc. con el fin de gestionar el trabajo enviado a cada una de las instancias. Amazon Cloud Watch[2] es una herramienta de monitorización para instancias de Amazon EC2 y ofrece gratuitamente medir 7 parámetros del rendimiento cada 5 minutos.

   En nuestro caso nos interesa medir: El rendimiento del CPU, memoria y el tráfico de red de cada instancia.

Para analizar posibles cuellos de botella en el tráfico nos interesa analizar la transmisión de la imagen de 1MB de tamaño puesto que el resto de transmisiones representan un porcentaje despreciable. Obsérvese que a pesar de que el tamaño del mensaje puede llegar a ser considerable, está transmisión en raras ocasiones (cuando se encuentre un mensaje oculto) por lo que podemos excluirla del análisis.

Analicemos primero el tráfico de datos en el nodo local. Usando una herramienta llamada Speed Test para medir la velocidad máxima de transferencia disponible[18] tenemos que el máximo de download/upload es de 3.9Mbps/2Mbps. Hemos visto que el tamaño de la información descargada no ocurre con frecuencia, por tanto, no se considerará en este análisis. En cambio el tamaño de los ficheros que hay que subir a la red no es despreciable. Teniendo en cuenta el tamaño de la imagen (Puede variar, en el ejemplo consideraremos 1MB) y con una velocidad de subida de 2Mbps tardaremos 4seg/MB en subir datos lo que significa que tardaremos 4 segundos en mandar la imagen a cada instancia.

Hemos analizado la transmisión de datos entre el nodo local y el nodo maestro del cloud provider. Como se observa en la Figura 4.3, ahora tenemos que analizar la transmisión de datos dentro del cloud provider. Observase que, cuantas más instancias esclavas tenga una instancia dentro de la jerarquía de árbol, mayor será la transmisión de datos a realizar

porque mayor será el número de veces que transmita la imagen a instancias esclavas bajo su control.

Este análisis del tráfico de datos que hemos hecho nos lleva a pensar que la red de interconexión se puede convertir en un cuello de botella y que conviene aplicar algún método para reducir el tráfico. En este trabajo la decisión sobre el tipo de instancias utilizadas y la arquitectura están encaminadas a ello.

Análisis de uso de CPU y Memoria

Al ejecutar el algoritmo en Matlab 2011a obtenemos que el nodo local usa el 100 % del CPU (@2,4GHz) y 170MB de memoria RAM. Como vemos, el programa no usa apenas memoria RAM y sin embargo, usa todo el CPU disponible. Por tanto, lo lógico es decantarse por instancias del tipo high CPU.

Una instancia media de CPU alta tiene (2 núcleos @ 2,5GHz, 1,7GB RAM y **Rendimiento E/S: Moderado**) mientras que una instancia extra grande de CPU alta tiene (8 núcleos @ 2,5GHz, 7GB RAM y **Rendimiento E/S: Alto**). Con 8 núcleos y 8 ejecutables tendríamos un uso de 1.3GB de memoria. Por lo tanto, dado que el rendimiento E/S de la instancia extra grande de CPU es alta, esta parece la mejor opción porque según la ecuación 4.2 se requerirá un menor número de instancias y como cada instancia solo requiere una copia de la imagen a analizar, el tráfico de datos generado también será menor evitando problemas de tráfico de datos.

4.3.2. Decisión sobre la arquitectura utilizada

En esta sección analizaremos las ventajas de utilizar una arquitectura con estructura master-slave de varios niveles como posibilidad para reducir la carga en la transmisión de datos.

La arquitectura de la que hemos partido para resolver este problema es de tipo master-slave como la que se ve en la Figura 4.3.

Las instancias master y semi-master son instancias de Amazon EC2 que se dedican a repartir el trabajo entre los slaves, monitorizar sus tiempos de ejecución, elegir el número óptimo de slaves, mandar el programa y los datos de entrada a los slaves, recoger los resultados y mandar los resultados relevantes al nodo local.

Los slaves son también instancias de Amazon EC2 que se dedican a ejecutar el programa y mandar los resultados y información sobre la ejecución al master sin decidir sobre la escalabilidad del problema.

Cada instancia slave analiza cada vez una opción de esteganoanálisis de entre todas las que se han podido usar para ocultar información como hemos visto en la sección 4.2.

1. **Estructura Master-slave simple:** En la Figura 4.4 se muestra la arquitectura master-slave de un nivel. Esta arquitectura es muy simple, sin embargo, como veremos más adelante, esta arquitectura tiene un problema: La red de interconexión creará un cuello de botella como ya habíamos augurado en la sección anterior.

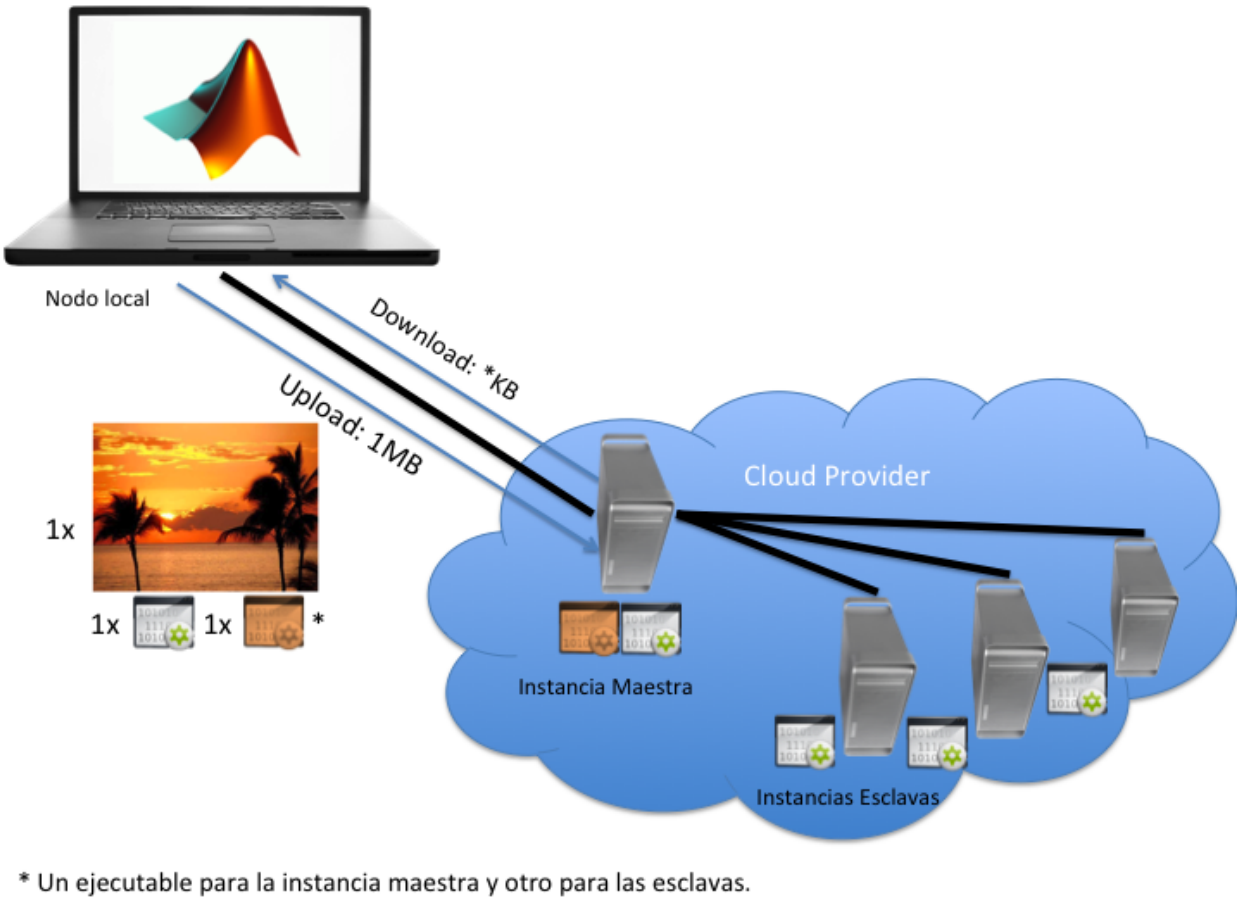

Figura 4.3: *Arquitectura básica del sistema utilizado*

Veamos que ocurre con un ejemplo. Separamos en dos partes el tráfico de datos que genera cada instancia:

a) **Tráfico de datos que se genera solo una vez:** Dentro de este grupo tenemos el programa ejecutable. Como solo se realiza una vez, no lo consideraremos a la hora de analizar el posible cuello de botella.

b) **Tráfico de datos que se repite para cada imagen analizada:** Dentro de este grupo tenemos los datos del programa, la imagen a analizar, la parte del esteganoanálisis a realizar y los resultados. La única parte de este grupo que genera un tráfico significativo es la imagen a analizar.

Consideremos la imagen de 1MB que hemos usado en este proyecto. Solo para analizar las distintas secciones desde 5x5 hasta 10x10 de forma paralela necesitamos 36 procesadores. Considerando una red de 5Mb/s de subida, la imagen tardaría 1,6s en enviarse. Si como hemos simulado en el capítulo anterior la iteración más lenta tarda

19s en ejecutarse (5x5) tendremos que una instancia maestra solo puede tener un máximo de 7 instancias esclavas porque de lo contrario se tardará más en la transmisión de la información que en ejecución.

Ejemplo: si tenemos 36 instancias organizadas como en la Figura 4.4 tendríamos 35 instancias esclavas por lo que el nodo maestro tardaría 35*1,6s=56s en transmitir la imagen a cada una de las instancias y la instancia que más tarda en procesar los datos tardaba 19s. Por lo tanto, todas las instancias estarían un mínimo de 37s inactivas.

Por tanto concluimos que esta arquitectura no es escalable porque en un punto siempre satura las interconexiones

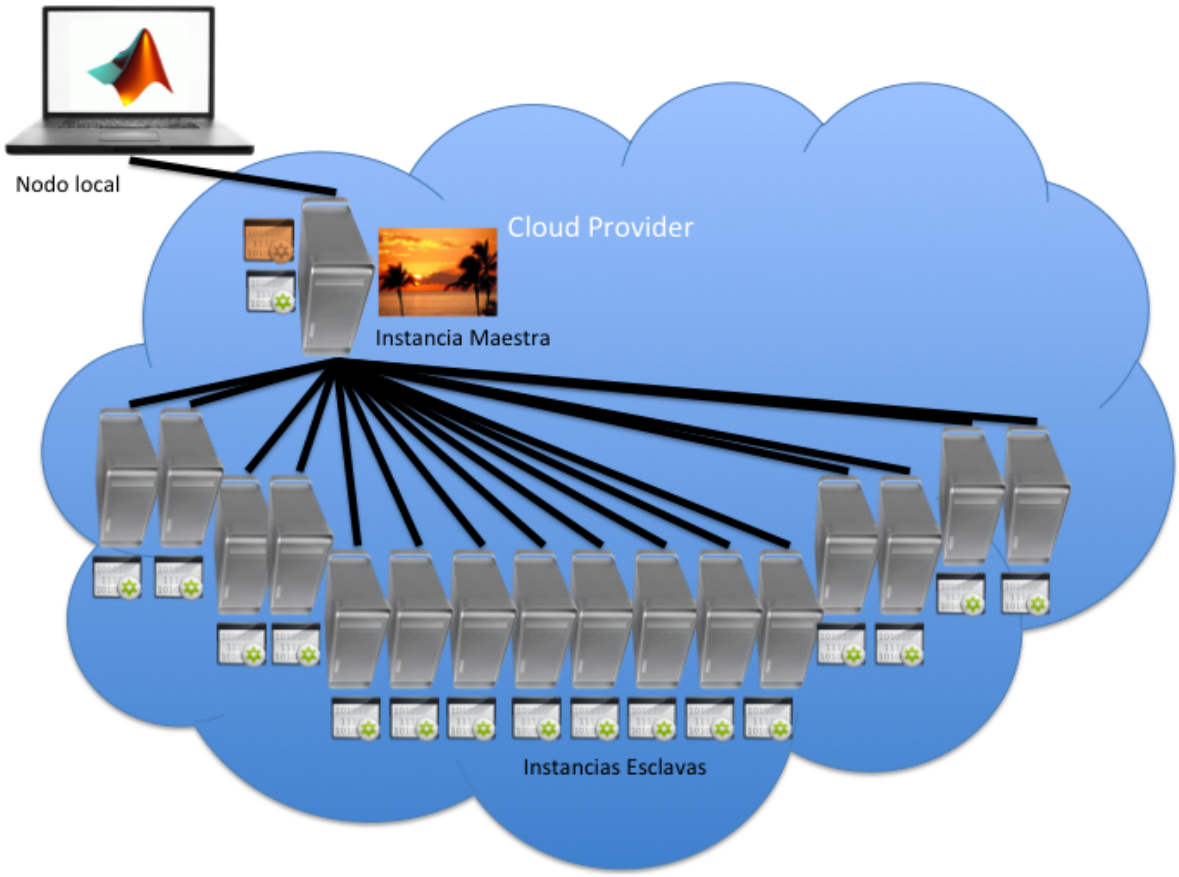

Figura 4.4: *Esta estructura no es escalable porque las conexiones de red producen un cuello de botella*

2. **Estructura master-slave de varios niveles:** Esta nueva arquitectura introduce un nivel intermedio entre las instancias master y las instancias slave como vemos en la Figura 4.5. Las instancias que están en este nivel intermedio se denominarán a partir de ahora instancias semi-maestras y su objetivo es administrar las instancias slave

que están bajo su control y transmitir al nodo master la información acerca de las instancias bajo su control y de si misma.

Retomamos el ejemplo anterior y consideremos ahora una instancia maestra cinco semi-maestras y 29 esclavas con la configuración (1 Maestra), (5 Semi-maestras) y (6,6,6,6,5 Esclavas). Si analizamos las transmisiones tendríamos 1,6s*5=8s desde la maestra a las semi-maestras y 1,6s*6= 9.6s entre las semi-maestras y las esclavas con un total de 17.6s en todas las transmisiones. Como vemos, esta arquitectura escala mucho mejor que la anterior.

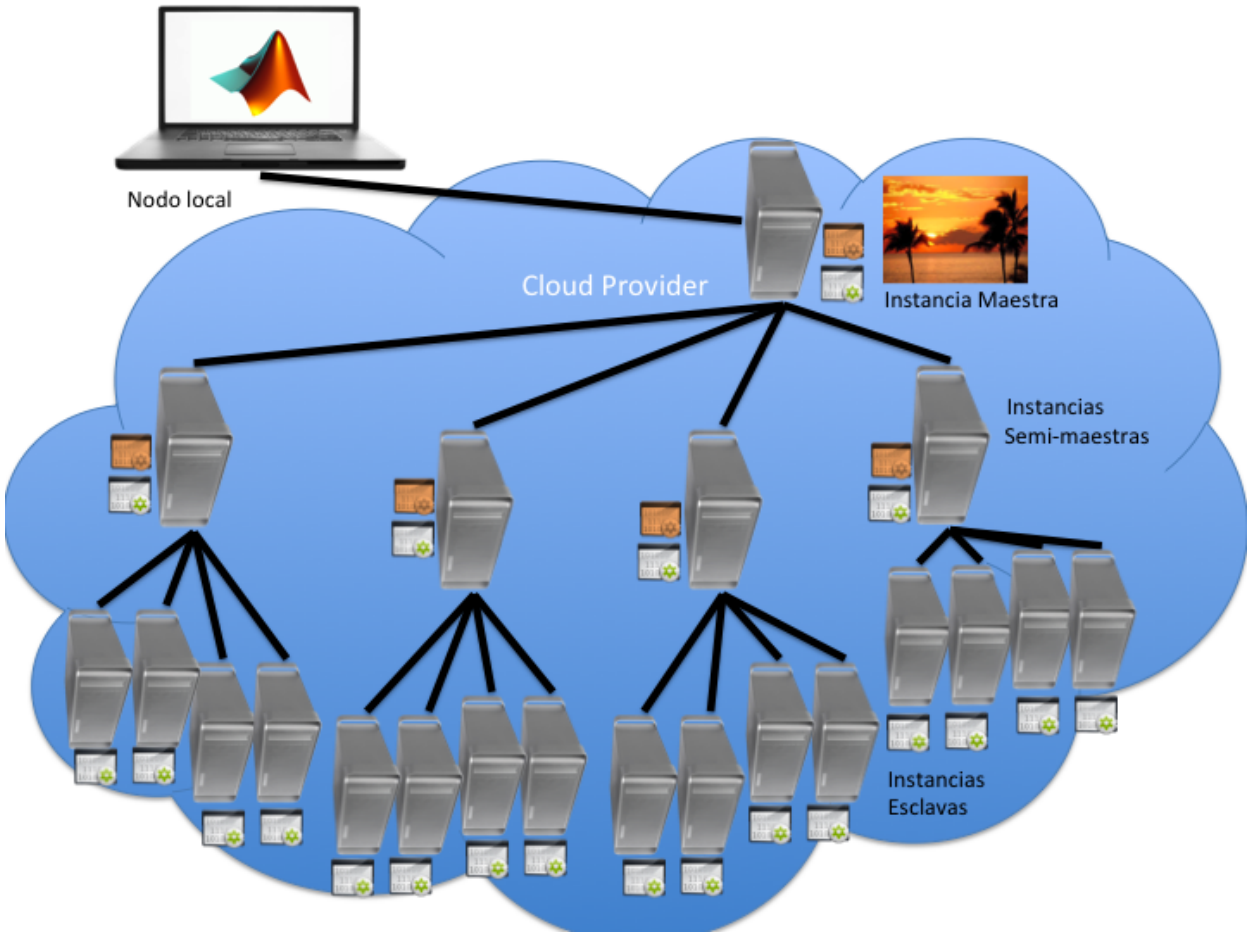

Figura 4.5: *Esta estructura si es escalable porque la conexiones de red están pensadas para reducir el tráfico de datos*

4.3.3. Auto-escalado de la arquitectura

Recordemos que el problema que planteamos es analizar las fotos de una red social on-line por lo que el algoritmo debe adaptarse en tiempo de ejecución a la cantidad y complejidad de las imágenes que se están analizando en ese momento. Es por tanto un problema dinámico puesto que el flujo de imágenes depende de los usuarios de la red social y por tanto, cambia constantemente y por ello plantearemos una arquitectura auto-escalable.

El análisis de las imágenes no siempre requiere el mismo computo debido principalmente a su distinto tamaño. Por ello, es necesario monitorizar el comportamiento y carga de trabajo de las instancias para ver si las predicciones son precisas o si las instancias están demasiado saturadas u ociosas y adaptar las instancias a la carga de trabajo actual con estas mediciones.

El algoritmo que hemos desarrollado es capaz de usar más o menos instancias en función de el número de imágenes a analizar, las pruebas a realizar sobre cada imagen y la fecha para la que tienen que estar analizadas todas las imágenes.

Predicción del número de instancias necesarias

Para hacer una predicción del número de instancias necesarias lo que hay que analizar es cuantos tipos de esteganografía se quieren analizar (considerando: secciones, transformadas), cuanto se tarda en analizar cada uno de media, cuantas posibles imágenes anfitrión se quieren analizar y para cuando deben estar analizadas.

Para hacer estas predicciones obtenemos las estadísticas de un experimento similar al de la tabla 3.1 pero con tamaños de sección que van desde 3x3 hasta 20x20. En la Figura 4.7, vemos el histograma de tiempos de análisis para el análisis de secciones de 3x3 a 20x20 en una imagen de 1MB y el mismo análisis repetido para una imagen de 5MB

De los resultados obtenidos en la ejecución en serie destacamos que el algoritmo requiere 1578 seg (26 minutos) para analizar 289 posibles formas de ocultar un mensaje en la imagen de 1MB y 9345 seg (155 minutos) en analizar lo mismo en una imagen de 5MB.

Vemos por tanto, que resulta imprescindible acelerar el algoritmo. Como hemos visto en la sección 4.3.1, cloud computing nos permite acelerar el proceso hasta el tiempo que tarda la iteración más lenta en ejecutarse que es de 29 segundos para el caso de la imagen de 1MB y 167 segundos para la imagen de 5MB. Obtenemos por tanto una aceleración de 55x y 56x respectivamente.

Lo que ahora nos interesa, es pronosticar el número de instancias necesarias para analizar una imagen antes de cierto momento prefijado. Obviamente si el tiempo restante es inferior al tiempo que tarda la iteración más lenta en ejecutarse no habrá forma de escalar el algoritmo para que los resultados estén a tiempo. Para resolver este problema analizaremos la siguiente imagen.

Esta figura, nos muestra el tiempo de analisis de las iteraciones normalizada a la iteración más rápida. Como vemos, la iteración más lenta tarda siempre aproximadamente 14 veces más que la iteración más rápida. Por tanto, para predecir cuanto tiempo tardará la iteración más larga en ejecutarse nos sirve con ejecutar primero la iteración más corta que es la iteración que analiza los bloques más grandes de 20x20 y multiplicar el tiempo obtenido por

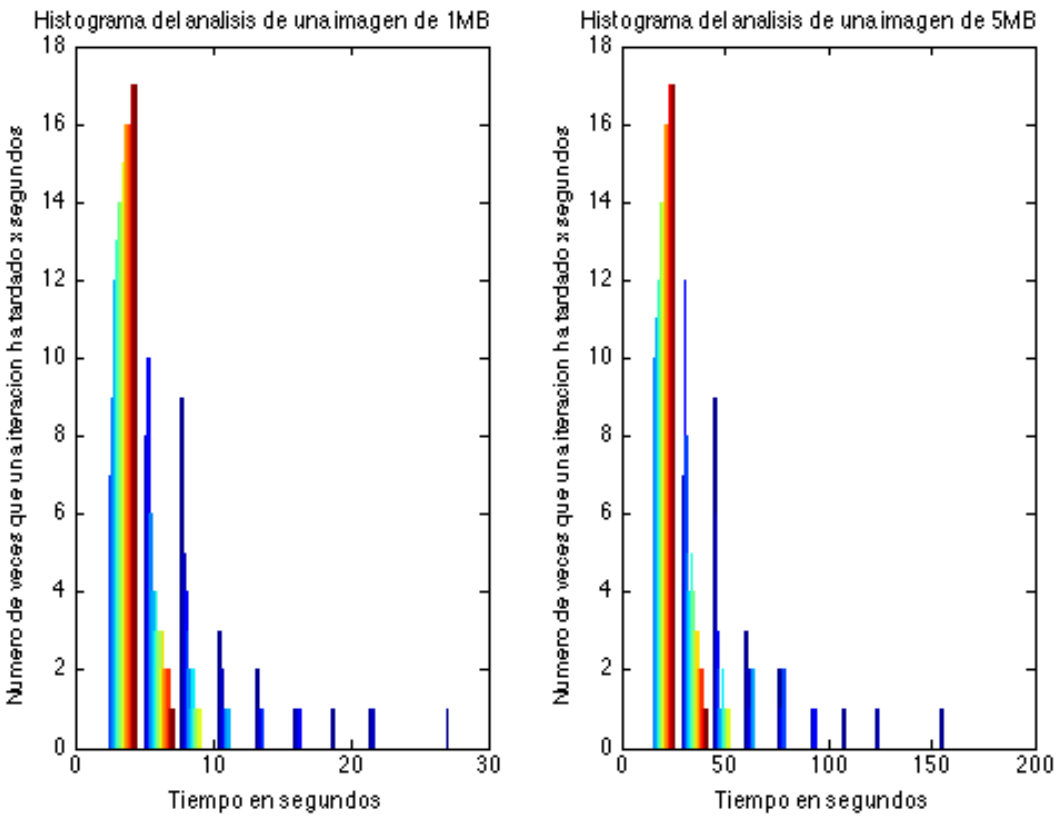

Figura 4.6: *Esta figura muestra el histograma de los tiempos de análisis del algoritmo, al analizar secciones de 3x3 a 20x20 para una imagen de 1MB y otra de 5MB*

15 para estimar cuanto tardara la iteración más lenta, con un margen de salvaguarda.

Una vez que hemos estimado cuanto tardará la iteración más lenta, nos falta estimar cual es el número mínimo de instancias necesarias para analizar las imágenes en ese periodo. Para ello calculamos la media de los histogramas anteriores con respecto a la iteración más rápida. También esta vez, los resultados siguen un patrón y para ambas imágenes el tiempo medio es aproximadamente 2,5 veces más lento que la iteracion más rápida y 5.6 veces más rápida que la iteración más lenta.

Por tanto, si usamos la salvaguarda de considerar que es 5 veces más lento que la iteración más larga y lo aplicamos a la Ecuación 4.3.1 tenemos que con floor(289/5)+1=58 instancias deberíamos ser capaces de analizar el algoritmo en el tiempo mínimo.

A partir de este punto, el análisis de como variar las instancias para que una imagen esté analizada para un momento dado es sencillo, aplicando la siguiente ecuación.

$$Num_{NewInstances} = floor(58 * 15 * IteracionMasCorta/TiempoRestanteEnSegundos) + 1 \quad (4.3)$$

Se explicará el uso de esta ecuación con el siguiente ejemplo:

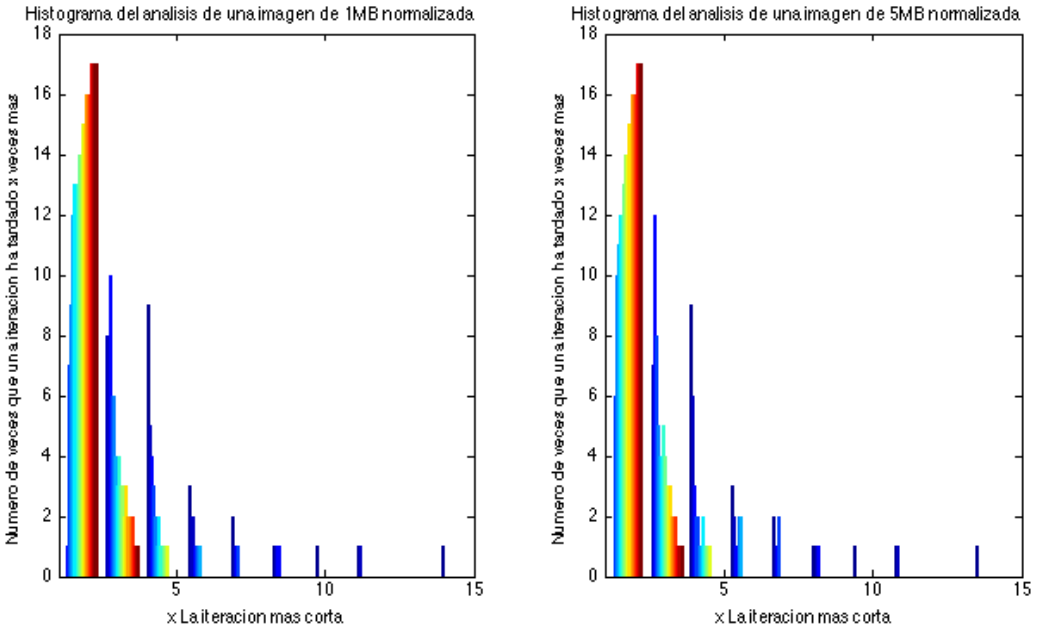

Figura 4.7: *Esta figura muestra el histograma de los tiempos de análisis normalizados (a la iteración más rapida) del algoritmo, al analizar secciones de 3x3 a 20x20 para una imagen de 1MB y otra de 5MB.*

Ejemplo: Considérese que hay que entregar un informe sobre la seguridad en las redes sociales a las 12:00 lo más completo posible. A las 11:00 se detecta una posible imagen sospechosa de 5MB, la iteración más rápida (bloques de 20x20) se realiza en 15 segundos.

Con este dato, estimamos que la iteración más lenta tardará 15x15=225 segundos y sabemos que para realizarlo en este tiempo necesitamos 58 instancias. Por tanto aplicando la ecuación:

$$Num_{NewInstances} = floor(58 * 225/3600) + 1 = 4 \tag{4.4}$$

Tenemos que con 4 instancias tendremos el resultado a la hora prevista. Si a las 11:15 se detecta otra posible imagen sospechosa de 10MB y la iteración más rápida se realiza en 35 segundos, tendríamos una estimación de la iteracion más lenta de 35x15=525 segundos y aplicando la misma ecuación:

$$Num_{NewInstances} = floor(58 * 525/2700) + 1 = 12 \tag{4.5}$$

Tendríamos que son necesarias 12 instancias más para obtener el resultado.

Obsérvese que el análisis del número mínimo de instancias es probabilístico. El programa, hace una predicción del número mínimo de instancias necesarias para analizar un flujo de imágenes variable. Es posible que en un momento dado, el número de instancias necesarias sea superior a la estimada y haya que arrancar nuevas máquinas. También es posible, que

el algoritmo encuentre algún mensaje oculto antes de terminar de ejecutarse y su financiación no sea necesaria. En este caso, hay una serie de recursos que quedan libres y deben descontarse al número de nuevas instancias solicitadas en el caso de que llegue una nueva imagen. Todo esto, debe ser controlado por la instancia master mandando en todo momento la información del proceso al nodo local.

Las características del auto-escalado implementado son las siguientes: Es un auto-escalado activo que predice el número de instancias necesarias por medio de una fórmula y actúa en consecuencia y escala los recursos de forma horizontal ya que arranca o apaga instancias y no cambia los recursos asignados a una instancia.

4.3.4. Experimento en Amazon EC2

Para obtener los resultados anteriores se ha realizado un experimento en Amazon EC2 siguiendo los pasos que se van a detallar.

Tras los pasos iniciales para usar Amazon EC2 (registro, login,...) estándar. Los pasos aquí expuestos se ejecutan con un script que configura la máquina de forma automática.

1. A la hora de lanzar la instancia, se elige el AMI ami-dcf615b5 de 32 bits / AMI ami-f061599 de 64 bits que se trata de la distribución Linux Debian

2. Se logea en la instancia con el comando:

   ssh -i .ec2/sana.pem root@ec2-184-72-145-24.compute-1.amazonaws.com

3. Se procede a instalar el software Octave 3.0:

   apt-get update

   aptitude install octave3.0

4. Se instala un plugin a Octave para entrada y salida de datos:

   apt-cache search octave io

   aptitude install octave io

5. Se inicia octave

   octave

6. Se envían los ficheros necesarios para la ejecución del programa:

   scp -i .ec2/sana.pem /Users/inigosananiceto/Dropbox/DOCTORADO/Amazon/AnalisisAmazonv1.
   root@ec2-184-72-145-24.compute-1.amazonaws.com:/home

   scp -i .ec2/sana.pem /Users/inigosananiceto/Dropbox/DOCTORADO/Amazon/Matrices.mat
   root@ec2-184-72-145-24.compute-1.amazonaws.com:/home

7. Se ejecuta el algoritmo y se hacen las mediciones

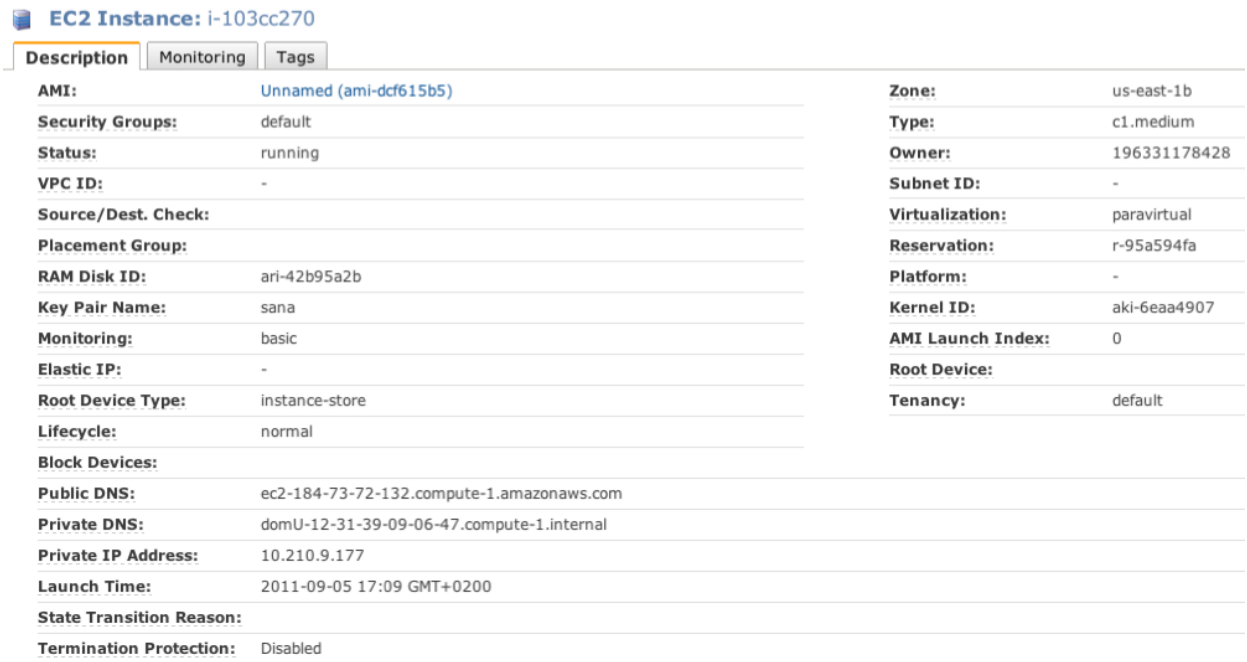

Figura 4.8: *Características de la instancia EC2*

4.3.5. Código de Octave

El código Octave ejecutado en la máquina de Amazon se añade en el apartado A.3. del Apéndice A.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

En este trabajo hemos desarrollado un sistema de esteganografía que esconde mensajes ocultos en imágenes y un sistema de esteganoanálisis que trata de hallar los mensajes ocultos en imágenes.

Para acelerar el proceso, hemos considerado diversos factores del algoritmo: paralelización posible; utilización de los recursos: CPU, RAM, Red, etc. Y hemos concluido que la arquitectura cloud que mejor se adapta a las necesidades del algoritmo es una arquitectura de cloud computing master-slave de varios niveles y auto-escalado inteligente con instancias high-CPU.

La decisión de utilizar una arquitectura de master-slave de varios niveles, es consecuencia de la gran carga de red que soportan las instancias maestras. Esta gran carga en la red, se debe a que las instancias maestras tienen que mandar imágenes con cierta frecuencia a todos las instancias slaves que están a su cargo. Con una arquitectura master-slave de varios niveles, se reduce en número de instancias slave que tiene una instancia maestra a su cargo minimizando así la carga de red y evitando un colapso de las comunicaciones.

La decisión de usar instancias high-CPU, es consecuencia de analizar el rendimiento de una máquina con una iteración del algoritmo y observar que usaba el 100 % del CPU y una ínfima parte de la memoria. Esto era de esperar por el tipo de aplicación implementada. De esta forma, las instancias high-CPU que ofrecen en comparación más CPU que RAM, son las instancias óptimas para ejecutar el algoritmo con el mínimo coste. Además, como vemos en la ecuación 4.2, el mayor número de cores en esta instancia nos permite disminuir el número de instancias necesarias y con ello, reducimos también el tráfico de la red, que como hemos visto, puede suponer un problema.

Finalmente, el auto-escalado. En la Tabla 3.2 vemos como analizar tamaños de sección más grandes, requiere menos tiempo que analizar tamaños de sección más pequeños. Por lo tanto, el algoritmo de auto-escalado ejecuta primero el análisis de las secciones más grandes de todas las imágenes que tiene pendientes de analizar y con ello, calcula cuanto le costará ejecutar el análisis de las secciones más pequeñas que son las más lentas. Con este

cálculo y según lo que hemos visto en 4.3.1, obtiene el mínimo tiempo en el que una imagen puede ser analizada. Teniendo en cuenta estos tiempos para todas las imágenes y el deadline establecido por el usuario, el algoritmo decide como auto-escalar siguiendo las ecuaciones obtenidas en la sección 4.3.3.

5.2. Trabajo futuro

En este trabajo se ha estudiado el auto-escalado de una aplicación ejecutada en un proveedor cloud. La inteligencia del auto-escalado, ha sido implementada en la misma aplicación usando parámetros internos para realizar predicciones. Sin embargo, como hemos visto en la sección 4.2.5 esto supone una carga extra para el programador de la aplicación y un problema para portar la aplicación a otros proveedores cloud.

Como trabajo futuro se pretende crear un broker que interaccione con el proveedor cloud y la aplicación para que la inteligencia del auto-escalado se implemente en broker y nos permita auto-escalar sin las limitaciones del proveedor cloud y sin una carga extra para el programador de la aplicación.

Bibliografía

[1] Amazon auto scaling web page, http://aws.amazon.com/autoscaling/, Aug 2011.

[2] Amazon cloud watch, http://aws.amazon.com/cloudwatch/, June 2011.

[3] Amazon ec2 web page, http://aws.amazon.com/ec2/, Aug 2011.

[4] Amazon instance types web page, http://aws.amazon.com/ec2/instance-types/, Aug 2011.

[5] Amazon pricing web page, http://aws.amazon.com/ec2/pricing, May 2011.

[6] Azure web page, http://www.microsoft.com/windowsazure/, Aug 2011.

[7] Cloud sigma pricing web page, http://www.cloudsigma.com/en/pricing/price-schedules, May 2011.

[8] Cloud sigma web page, http://www.cloudsigma.com/, Aug 2011.

[9] Elastichost pricing web page, http://www.elastichosts.com/cloud-hosting/pricing, May 2011.

[10] Globustoolkit web page, http://www.globus.org/toolkit/, Sep 2011.

[11] Gogrid pricing web page, http://www.gogrid.com/cloud-hosting/cloud-hosting-pricing.php, May 2011.

[12] Gogrid web page, http://www.gogrid.com/, Aug 2011.

[13] Google app engine web page, http://code.google.com/appengine/, Aug 2011.

[14] Net suite web page, http://www.netsuite.com/portal/home.shtml, Aug 2011.

[15] Octave web page, http://www.gnu.org/software/octave/, Sep 2011.

[16] Oracle crm on demand web page, http://crmondemand.oracle.com/es/index.html, Aug 2011.

[17] Salesforce web page, http://www.salesforce.com/es/platform/, Aug 2011.

[18] Speed test web page, http://www.speedtest.net/, Aug 2011.

[19] Zaidoon Kh. Al-Ani, A. A. Zaidan, B. B. Zaidan, and Hamdan O. Alanazi. Overview: Main fundamentals for steganography. *CoRR*, abs/1003.4086, 2010.

[20] R.J Anderson and F.A.P Petitcolas. On the limits of steganography. *Selected Areas in Communications, IEEE Journal on*, 16(4):474–481, 1998.

[21] Iñigo San Aniceto, Rafael Moreno-Vozmediano, Ruben S.Montero, and Ignacio M.Llorente. Cloud capacity reservation for optimal service deployment. *Cloud Computing International Conference*, 1:8, sep 2011.

[22] M Armbrust, A Fox, R Griffith, A.D Joseph, R.H Katz, A Konwinski, G Lee, D.A Patterson, A Rabkin, and I Stoica. Above the clouds: A berkeley view of cloud computing. Technical report, EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[23] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53:50–58, April 2010.

[24] Afkham Azeez. Auto-scaling Web Services on Amazon EC2. *University of Moratuwa*, pages 1–8, July 2008.

[25] H. B. Bahar and Ali Aboutalebi. Image steganography, a new approach for transferring security information. *CoRR*, abs/0808.1410, 2008.

[26] LFE Barrios. Cloud computing como una red de servicios. Technical report, Instituto Técnologico de Costa Rica, November 2009.

[27] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616, 2009.

[28] A.G.H. Chamorro and M.N. Miyatake. A new methodology of image steganalysis including for jpeg steganography. In *Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2010*, pages 434 –438, 28 2010-oct. 1 2010.

[29] R Chandramouli. A Mathematical Approach to Steganalysis. In *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV, California*, Jan 2002.

[30] Marcos de Assunção, Alexandre di Costanzo, and Rajkumar Buyya. A cost-benefit analysis of using cloud computing to extend the capacity of clusters. *Cluster Computing*, 13:335–347, sep 2010.

[31] Mahmoud Elnajjar, A. A. Zaidan, B. B. Zaidan, Mohamed Elhadi M. Sharif, and Hamdan O. Alanazi. Optimization digital image watermarking technique for patent protection. *CoRR*, abs/1002.4049, 2010.

[32] Ian Foster. What is the Grid? A Three Point Checklist. *Argonne National Laboratory and University of Chicago*, 1(6), 2002.

[33] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *2008 Grid Computing Environments Workshop*, pages 1–10. IEEE, 2008.

[34] Rolf Harms and Michael Yamrtino. EU Public Sector Cloud Economics. Technical report, Microsoft, 2011.

[35] A. Hernandez-Chamorro, A. Espejel-Trujillo, J. Lopez-Hernandez, M. Nakano-Miyatake, and H. Perez-Meana. A methodology of steganalysis for images. In *Electrical, Communications, and Computers, 2009. CONIELECOMP 2009. International Conference on*, pages 102 –106, feb. 2009.

[36] Rafiqul Islam, A. W. Naji, A. A. Zaidan, and B. B. Zaidan. New system for secure cover file of hidden data in the image page within executable file using statistical steganography techniques. *CoRR*, abs/1002.2416, 2010.

[37] HA Jalab and AA Zaidan. New Design for Information Hiding with in Steganography Using Distortion Techniques. *whitepapers.hackerjournals.com*.

[38] Yang Jie, Qiu Jie, and Li Ying. A profile-based approach to just-in-time scalability for cloud applications. In *Cloud Computing, 2009. CLOUD '09. IEEE International Conference on*, pages 9 –16, sept. 2009.

[39] Gang Luo, Xing ming Sun, Ling yun Xiang, and Jun wei Huang. An evaluation scheme for steganalysis-proof ability of steganographic algorithms. In *Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. Third International Conference on*, volume 2, pages 126 –129, nov. 2007.

[40] M Mao, J Li, and Marty Humphrey. Cloud Auto-Scaling with Deadline and Budget Constraints. *Conference on Grid Computing*, oct 2010.

[41] Ming Mao, Jie Li, and M. Humphrey. Cloud auto-scaling with deadline and budget constraints. In *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pages 41 –48, oct. 2010.

[42] Alfonso Muñoz Muñoz, Justo Carracedo Gallardo, and Sergio Sánchez García. Detection of distributed steganographic information in social networks. In *Proceedings of the 2008 Euro American Conference on Telematics and Information Systems*, EATIS '08, pages 10:1–10:9, New York, NY, USA, 2008. ACM.

[43] Samir B. Patel, Shrikant N. Pradhan, and Saumitra U. Ambegaokar. A novel approach for implementing steganography with computing power obtained by combining cuda and matlab. *CoRR*, abs/0912.0947, 2009.

[44] N Provos and P Honeyman. Hide and seek: an introduction to steganography. *Security & Privacy, IEEE*, 1(3):32–44, 2003.

[45] Sabu M. Thampi. Information hiding techniques: A tutorial review. *CoRR*, abs/0802.3746, 2008.

[46] R. Uhlig, G. Neiger, D. Rodgers, A.L. Santoni, F.C.M. Martins, A.V. Anderson, S.M. Bennett, A. Kagi, F.H. Leung, and L. Smith. Intel virtualization technology. *Computer*, 38(5):48 – 56, may 2005.

[47] Luis M. Vaquero, Luis Rodero-Merino, and Rajkumar Buyya. Dynamically scaling applications in the cloud. *SIGCOMM Comput. Commun. Rev.*, 41:45–52.

[48] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1:7–18, may 2010.

Apéndice A

Códigos de Matlab y Octave

A.1. Código Matlab del sistema de esteganografía

1. La llamada a las funciones
2. La función de transmisión
3. La función de recepción
4. La función que secciona la imagen
5. La función que junta la imagen

1. Llamada a las funciones

```
function Matrices=FFTiFFTv14(Matrices,mensaje_bin,Paths)
% Se llama a la función de Transmisión
Matrices=Esteganografia(Matrices, mensaje_bin,Paths);
% Se llama a la función de Recepción
Matrices=RecepcionEsteganografia(Paths);
save([Paths.SavePath,'Matrices.mat'],'Matrices');
```

2. Función de transmisión

```
function Matrices=Esteganografia(Matrices, mensaje_bin,Paths);
% Divido la imagen en Secciones de 8*8 llamando a BloquesSecv0
Matrices=BloquesSecv0(Matrices,0);
% Obtengo dimensiones de la imagen seccionada y n el número de secciones
[alto,ancho,fondo]=size(Matrices.MatrizImagen);
sec_ancho=floor(ancho/8); sec_alto=floor(alto/8);sec_fondo=fondo;
[caracteres, bits]=size(mensaje_bin);
MatrizSecciones=Matrices.MatrizSecciones;
% Inicio el bucle que recorre las secciones y meto el mensaje
for seck=1:sec_fondo %RGB
    for secj=1:sec_alto %Alto
        for seci=1:sec_ancho %Ancho
        % Compruebo que no meto mas secciones de las que tengo.
            if (((seck-1)*sec_ancho*sec_alto)+((secj-1)*sec_ancho)+seci)<=caracteres
% Transformada de Fourier del vector de la sección
                freq_sec_lineal=fft(Matrices.MatrizSecciones(secj,seci,seck,:)); % Discrete Fourier transform
                freq_sec_lineal_bin1=dec2bin(65536+real(freq_sec_lineal)); % Parte Real: +65536 offset
```

```
                freq_sec_lineal_bin2=dec2bin(65536+imag(freq_sec_lineal)); % Parte Imaginaria: +65536 offset
                [filas,columnas]=size(freq_sec_lineal_bin1);
  % Introduzco información de la longitud del mensaje
                if (seck==1 && secj==1 && seci==1)
                    figure
                    VectorFreq1(1,:)=freq_sec_lineal(1,1,1,:);
                    plot(VectorFreq1);
                    title('Vector de frecuencias de la seccion 1x1x1');
                    xlabel('Valor del pixel');
                    ylabel('Posicion del pixel dentro del vector');
                    figure
                    VectorNat(1,:)=Matrices.MatrizSecciones(1,1,1,:)
                    plot(VectorNat)
                    title('Vector con los valores de los pixeles en la primera seccion');
                    xlabel('Valor del pixel');
                    ylabel('Posicion del pixel dentro del vector');
                    bincaracteres=dec2bin(caracteres,14); %14 bits.
                    freq_sec_lineal_bin1(1:2*bits,columnas)=bincaracteres;
                else
                    % El -1 viene de que en la primera sección se ha codificado la longitud del mensaje
                    % Estenografiar ultimo bit de la parte real
                    freq_sec_lineal_bin1(1:bits,columnas)=mensaje_bin(((secj-1)*sec_ancho)+seci-1,1:bits);
                end
% Transformada Inversa de Fourier
% Numero Complejo: -65536 offset
                freq_sec_lineal_steg=complex(bin2dec(freq_sec_lineal_bin1)-65536,bin2dec(freq_sec_lineal_bin2)-65536);
                MatrizSecciones(secj,seci,seck,:)=ifft(freq_sec_lineal_steg,'symmetric'); % Inverse discrete Fourier transform
            end
        end
    end
end
disp('Mensaje introducido')
% Restablezco las secciones de 8*8 en la imagen
Matrices.MatrizSecciones=MatrizSecciones;
Matrices=BloquesJunv0(Matrices);
% Guardo la matriz imagen en un fichero .mat
save([Paths.SavePath,'Matrices.mat'],'Matrices');
```

3. Función de recepción

```
function Matrices=RecepcionEsteganografia(Paths)
% Carga los datos de la matriz imagen
load([Paths.SavePath,'Matrices.mat']);
% Divido la imagen en Secciones de 8*8 llamando a BloquesSecv0
Matrices=BloquesSecv0(Matrices,1);
% Obtengo dimensiones de la imagen seccionada
[alto,ancho,fondo]=size(Matrices.MatrizImagen);
sec_ancho=floor(ancho/8); sec_alto=floor(alto/8); sec_fondo=fondo;
% Inicio el bucle que recorre las secciones y obtengo el mensaje
mensajecompleto='';
bits=7;
caracteres=1000000;%Se actualizara
for seck=1:sec_fondo
    for secj=1:sec_alto
        for seci=1:sec_ancho
            if (((seck-1)*sec_ancho*sec_alto)+((secj-1)*sec_ancho)+seci)<=caracteres
% Trasformada de Fourier del vector de la sección
                freq_sec_analysis_lineal=fft(Matrices.MatrizSecciones(secj,seci,seck,:)); % Discrete fourier transform
                freq_sec_analysis_lineal_bin1=dec2bin(65536+real(freq_sec_analysis_lineal)); % Parte Real: +65536 offset
                [filas,columnas]=size(freq_sec_analysis_lineal_bin1);
                if (seck==1 && secj==1 && seci==1)
                    bincaracteres=freq_sec_analysis_lineal_bin1(1:2*bits,columnas);
                    caracteres=bin2dec(bincaracteres'); %Transpuesta
                else
```

```
% Obtengo el mensaje esteganografío con el bit menos significativo
                    mensaje_bin2(1:bits)=freq_sec_analysis_lineal_bin1(1:bits,columnas);
                    mensaje=bin2dec(mensaje_bin2);
                    mensajecompleto=[mensajecompleto, mensaje];
                    %Nota: No necesito hacer iFFT ni quitar el offset.}
                end
            end
        end
        mensajecompleto=[mensajecompleto,13];
    end
end
disp(['El mensaje en números es (100 caracteres): ',13,num2str(abs(mensajecompleto(1:100)))]);
disp(['El mensaje es (100 caracteres): ',13,char(mensajecompleto(1:100))]);
disp(['La longitud del mensaje es: ',num2str(length(mensajecompleto))]);
```

4. Función que secciona la imagen

```
function Matrices=BloquesSecv0(Matrices,steg)
% Ver si estoy seccionando una imagen original o esteganografiada
if steg==0
    MatrizImagen=Matrices.MatrizImagen;
else
    MatrizImagen=Matrices.MatrizImagenSteg;
end
[alto,ancho,fondo]=size(MatrizImagen);
% Defino el numero de secciones y creo la Matriz 4D y una variable para guardar las secciones
sec_ancho=floor(ancho/8); sec_alto=floor(alto/8);sec_fondo=fondo;
MatrizSecciones=zeros(sec_alto, sec_ancho,sec_fondo, 8*8); % De forma lineal para cada sección
S_secj_seci_seck=zeros(8,8,3);
% Inicio el bucle que recorre las la imagen y secciono en 8*8
for seck=1:sec_fondo
    for secj=1:sec_alto
        for seci=1:sec_ancho
% Consigo la matriz del bloque para cada color
            S_secj_seci_seck(:,:,seck)=MatrizImagen(((secj-1)*8)+1:((secj-1)*8)+8,((seci-1)*8)+1:((seci-1)*8)+8,seck);
% Vectorizo el bloque para que se pueda aplicar fft
            for j=1:8
                for i=1:8
                    MatrizSecciones(secj,seci,seck,((j-1)*8)+i)=S_secj_seci_seck(j,i,seck); % Recorre horizontalmente
                end
            end
        end
    end
end
% Añado la parte de imagen sobrante tras seccionar
MatrizSobrante=zeros(size(MatrizImagen));
MatrizSobrante(:,(sec_ancho*8)+1:ancho,:)=MatrizImagen(:,(sec_ancho*8)+1:ancho,:); % Lado derecho
MatrizSobrante((sec_alto*8)+1:alto,:,:)=MatrizImagen((sec_alto*8)+1:alto,:,:); % Abajo
% Compruebo que se ha seccionado bien
[y,x,z,v]=size(MatrizSecciones);
total1=sum(sum(sum(sum(MatrizSecciones))));
total2=sum(sum(sum(MatrizSobrante)));
if steg==0
    Matrices.SumaOrigSec=total1+total2;
else
    Matrices.SumaEstegSec=total1+total2;
end
disp(['Matriz secciones creada. Dimensiones: ',num2str(x),'*',num2str(y),'*',num2str(z),'.vector',num2str(v),'.']);
if(Matrices.SumaOrig==Matrices.SumaOrigSec && steg==0 || Matrices.SumaEsteg==Matrices.SumaEstegSec && steg==1 )
disp('Bien Seccionado')
%Muestro el vector de la primera sección antes de steg
    figure;
    subplot(1,2,1);
    Vector1(1,:)=MatrizSecciones(1,1,1,:);
```

```
    plot(Vector1);
    title('Vector de la seccion 1x1x1');
    xlabel('Value of pixel');
    ylabel('Posicion del vector');
else
    if steg==0
        disp(['Mal Seccionado orig, diferencia suma: ',num2str(abs(Matrices.SumaEsteg==Matrices.SumaEstegSec))])
    else
        disp(['Mal Seccionado esteg, diferencia suma: ',num2str(abs(Matrices.SumaOrig==Matrices.SumaOrigSec))])
    end
end
% Guardo la matriz en la estructura.}
Matrices.MatrizSobrante=MatrizSobrante;
Matrices.MatrizSecciones=MatrizSecciones;
```

5. Función que junta la imagen

```
function Matrices=BloquesJunv0(Matrices)
% Inicializo la nueva imagen
MatrizSecciones=Matrices.MatrizSecciones;
MatrizImagenSteg=zeros(size(Matrices.MatrizImagen));
% Inicio el bucle que recorre las secciones une la imagen
[sec_alto, sec_ancho,sec_fondo,vector]=size(MatrizSecciones);
for seck=1:sec_fondo
    for secj=1:sec_alto
        for seci=1:sec_ancho
            for j=1:8
                for i=1:8
                    MatrizImagenSteg(((secj-1)*8)+j,((seci-1)*8)+i,seck)=MatrizSecciones(secj,seci,seck,((j-1)*8)+i);
                end
            end
        end
    end
end
% Reintroduzco Matriz Sobrante
[alto,ancho,fondo]=size(Matrices.MatrizImagen);
MatrizImagenSteg(:,(sec_ancho*8)+1:ancho,:)=Matrices.MatrizSobrante(:,(sec_ancho*8)+1:ancho,:); %Lado derecho
MatrizImagenSteg((sec_alto*8)+1:alto,:,:)=Matrices.MatrizImagen((sec_alto*8)+1:alto,:,:); % Abajo
% Compruebo que se ha juntado bien
Matrices.MatrizImagenSteg=MatrizImagenSteg;
[y,x,z]=size(MatrizImagenSteg);
Matrices.SumaEsteg=sum(sum(sum(MatrizImagenSteg)));
disp(['Matriz Imagen Steg creada. Dimensiones: ',num2str(x),'*',num2str(y),'*',num2str(z),'.']);
disp(['Diferencia Matriz Steg con Original: ',num2str(abs(Matrices.SumaOrig-Matrices.SumaEsteg))])
```

A.2. Código Matlab del sistema de esteganoanálisis

1. La llamada a las funciones
2. Las funciones que obtienen estadísticas
3. Función de esteganoanálisis
4. Función que calcula cada iteración
5. Función que realiza la iteración

1. Llamada a las funciones

```
function Analisisv6(Matrices,Paths)
% Llamada a la función que calcula las estadísticas de la matriz imagen
Estadisticas.Original=CalcEstMatrizImagen(Matrices);
```

```
% Llamada a la función que calcula las estadísticas de la matriz esteganongrafiada
Estadisticas.Esteg=CalcEstMatrizSteg(Matrices);
% Llamada a la función que calcula las estadísticas de la diferencia
Estadisticas.Diferencia=CalcEstDiferencia(Estadisticas);
% Llamada a la función que muestro los resultados
Resultados(Estadisticas);
% Llamada a la función que hace el esteganoanálisis
Esteganoanalisisv0(Matrices,Paths);
```

2. Funciones que obtienen las estadísticas

```
% Función que obtienen las estadísticas de la imagen original
function Estadisticas=CalcEstMatrizImagen(Matrices)
% Transformar en vector para calcular estadísticas usando funciones de matlab
MatrizImagen=Matrices.MatrizImagen;
[alto,ancho]=size(MatrizImagen);
VectorImagen=zeros(1,alto*ancho);
for j=1:alto
    for i=1:ancho
        VectorImagen(1,((j-1)*ancho)+i)=MatrizImagen(j,i);
    end
end
% Obtener estadisticas
Estadisticas.MediaImagen=mean(VectorImagen);
Estadisticas.MedianaImagen=median(VectorImagen);
Estadisticas.VarianzaImagen=var(VectorImagen);
Estadisticas.CovarianzaImagen=cov(VectorImagen);
Estadisticas.StandardDeviationImagen=std(VectorImagen);
Estadisticas.VectorImagen=VectorImagen;
% Función que obtienen las estadísticas de la imagen esteganografiada
function Estadisticas=CalcEstMatrizSteg(Matrices)
% Transformar en vector para calcular estadísticas usando funciones de matlab
MatrizImagenSteg=Matrices.MatrizImagenSteg;
[alto,ancho]=size(MatrizImagenSteg);
VectorImagenSteg=zeros(1,alto*ancho);
for j=1:alto
    for i=1:ancho
        VectorImagenSteg(1,((j-1)*ancho)+i)=MatrizImagenSteg(j,i);
    end
end
% Obtener estadísticas
Estadisticas.MediaImagenSteg=mean(VectorImagenSteg);
Estadisticas.MedianaImagenSteg=median(VectorImagenSteg);
Estadisticas.VarianzaImagenSteg=var(VectorImagenSteg);
Estadisticas.CovarianzaImagenSteg=cov(VectorImagenSteg);
Estadisticas.StandardDeviationImagenSteg=std(VectorImagenSteg);
Estadisticas.VectorImagenSteg=VectorImagenSteg;
% Función que obtienen las estadísticas de la diferencia
function Estadisticas=CalcEstDiferencia(Estadisticas)
% Obtener los vectores
VectorImagen=Estadisticas.Original.VectorImagen;
VectorImagenSteg=Estadisticas.Esteg.VectorImagenSteg;
% Obtener estadisticas
VectorDiferencia=zeros(1,length(VectorImagen));
for i=1:length(VectorImagen)
    VectorDiferencia(1,i)=VectorImagenSteg(1,i)-VectorImagen(1,i);
end
% Obtener estadísticas
Estadisticas.MediaImagenDiferencia=mean(VectorDiferencia);
Estadisticas.MedianaImagenDiferencia=median(VectorDiferencia);
Estadisticas.VarianzaImagenDiferencia=var(VectorDiferencia);
Estadisticas.CovarianzaImagenDiferencia=cov(VectorDiferencia);
Estadisticas.StandardDeviationImagenDiferencia=std(VectorDiferencia);
Estadisticas.VectorImagenDiferencia=VectorDiferencia;
```

```
% Función que muestra los resultados
function Resultados(Estadisticas)
% Obtener las estadísticas
Original=Estadisticas.Original;
Esteg=Estadisticas.Esteg;
Diferencia=Estadisticas.Diferencia;
% Mostrar las estadísticas en modo texto
disp(['Media Imagen, ',num2str(Original.MediaImagen),' y Media Imagen Steg, ',
num2str(Esteg.MediaImagenSteg),' la diferencia es de: ',
num2str(abs((Original.MediaImagen-Esteg.MediaImagenSteg)/Original.MediaImagen)*100),'%']);
disp(['Mediana Imagen, ',num2str(Original.MedianaImagen),' y Mediana Imagen Steg, ',
num2str(Esteg.MedianaImagenSteg),' la diferencia es de: ',
num2str(abs((Original.MedianaImagen-Esteg.MedianaImagenSteg)/Original.MedianaImagen)*100),'%']);
disp(['Varianza Imagen, ',num2str(Original.VarianzaImagen),' y Media Imagen Steg, ',
num2str(Esteg.VarianzaImagenSteg),' la diferencia es de: ',
num2str(abs((Original.VarianzaImagen-Esteg.VarianzaImagenSteg)/Original.VarianzaImagen)*100),'%']);
disp(['Covarianza Imagen, ', num2str(Original.CovarianzaImagen),' y Covarianza Imagen Steg, ',
num2str(Esteg.CovarianzaImagenSteg),' la diferencia es de: ',
num2str(abs((Original.CovarianzaImagen-Esteg.CovarianzaImagenSteg)/Original.CovarianzaImagen)*100),'%']);
disp(['Standard Deviation Imagen, ', num2str(Original.StandardDeviationImagen),' y Standard Deviation Imagen Steg, ',
num2str(Esteg.StandardDeviationImagenSteg),' la diferencia es de: ',
num2str(abs((Original.StandardDeviationImagen-Esteg.StandardDeviationImagenSteg)/Original.StandardDeviationImagen)*100),'%']);
% Mostrar las estadísticas gráficamente
figure;
subplot(2,2,1);
hist(Original.VectorImagen,255);
title('Histogram of the image without steganography');
xlabel('Value of pixel');ylabel('Number of Times that value have been seen');
subplot(2,2,2);
hist(Esteg.VectorImagenSteg,255);
title('Histogram of the image with steganography');
xlabel('Value of pixel');ylabel('Number of Times that value have been seen');
subplot(2,2,3);
hist(Diferencia.VectorImagenDiferencia,10*255);
axis([-5,5,0,100])
title('Histogram of the diference');
xlabel('Value diference of pixel');ylabel('Number of Times that value have been seen');
```

3. Función de esteganoanálisis

```
function Matrices=Esteganoanalisisv0(Matrices,Paths);

inicio=tic;
% Funcion de probabilidad del espanol antiguo / ingles
VectorEsp=[12.2 1.5 3.6 5.3 14 0.5 1 1.2 5.5 0.6 0 5.4 2.7 6.9 9.9 2.2 2 6.2 7.7 3.8 4.8 1.1 0 0 1.5 0.4]; %Don quijote
%vectorhist=[12.53 1.42 4.68 5.68 13.68 0.69 1.01 0.7 6.25 0.44 0.01 4.97 3.15 6.71 8.68 2.51 0.88 6.87
7.98 4.63 3.93 0.9 0.02 0.22 0.9 0.52]; %Espanol
% Coloco la función de probabilidad en su lugar ASCII
Matrices.HistoEspPorc=zeros(1,128);
Matrices.HistoEspPorc(98:123)=VectorEsp(1:26)/100;
Matrices.m1=max(Matrices.HistoEspPorc(97:122));
% Empiezo el bucle anidado => Esto se paraleliza en la sección de cloud computing
imin=3;imax=20;
jmin=3;jmax=20;
iteraciones=(jmax-jmin+1)*(imax-imin+1);
for j=jmin:jmax
    for i=imin:imax
        iteracionporc=((i-imin+1)+((j-jmin)*(imax-imin+1)))/iteraciones;
        ResultadosAnalisis=AnalisisIteracionv0(Matrices,j,i);
        disp(['Completado en un: ',num2str(iteracionporc*100),'%']);
    end
end

save([Paths.SavePath,'ResultadosAnalisis.mat'],'ResultadosAnalisis');
```

```
fin=toc(inicio);
disp(['Tiempo total de esteganoanalisis: ',num2str(fin-inicio)]);
```

4. Función que calcula cada iteración

```
function ResultadosAnalisis=AnalisisIteracionv0(Matrices,j,i)
% Divido la imagen en Secciones de j*i}
inicioiteracion=tic;
Matrices=BloquesAnalisisv1(Matrices,j,i);
% Analisis en bloques i*j el numero de secciones en bloque análisis
mensajecompleto2='';
for seck=1:3
    for secj=1:Matrices.sec_alto
        for seci=1:Matrices.sec_ancho
% Trasformada de Fourier del vector de la sección
freq_sec_analysis_lineal=fft(Matrices.MatrizSeccionesVar(secj,seci,seck,:)); % Discrete fourier transform
            freq_sec_analysis_lineal_bin1=dec2bin(65536+real(freq_sec_analysis_lineal)); % Parte real: +65536 offset
            %freq_sec_analysis_lineal_bin2=dec2bin(65536+imag(freq_sec_analysis_lineal));
% Obtengo el mensaje esteganografio con el bit menos significativo}
            [filas,columnas]=size(freq_sec_analysis_lineal_bin1);
            mensaje2_bin2(1:7)=freq_sec_analysis_lineal_bin1(1:7,columnas);
            mensaje2=bin2dec(mensaje2_bin2);
            mensajecompleto2=[mensajecompleto2, mensaje2];
        end
        mensajecompleto2=[mensajecompleto2,13];
    end
end
% Normalizo las funciones para hacer la correlación
HistoMensaje=hist(abs(mensajecompleto2),128); % Comparare resultados con lenguaje, 128 ASCII
HistoMensajePorc=HistoMensaje/length(mensajecompleto2); m2=max(HistoMensajePorc(97:128));
HistoMensajeNorm=HistoMensajePorc*Matrices.m1/m2; % Normalizo
% Hallo la correlacion
coract=corr2( HistoMensajeNorm(97:122),Matrices.HistoEspPorc(97:122));
disp(['La correlacion es: ',num2str(coract)]);
% Si la correlación supera cierto limite, lo saco por pantalla
if coract>0.8
    disp(['Analizando bloques de: ',num2str(i),'*',num2str(j),' el mensaje es (100 caracteres): ',13,mensajecompleto2(1:100)]);
    figure;
    subplot(1,2,1);
    plot(HistoMensajeNorm);hold on;
    plot(Matrices.HistoEspPorc,'r'); hold off;
    title('Histogram of the image ASCII characters');
    xlabel('Value of ASCII character'); ylabel(['Frequencia con la que se ha visto ese caracter en bloques:
    (',num2str(j),',',num2str(i),')']);
    subplot(1,2,2);
    plot(HistoMensajeNorm(97:122));hold on;
    plot(Matrices.HistoEspPorc(97:122),'r'); hold off;
    title('Histograma del analisis de la imagen en ASCII comparado con el espanol antiguo');
    xlabel('Valor ASCII'); ylabel(['Frequencia con la que se ha visto ese caracter: (',num2str(j),',',num2str(i),')']);
    ResultadosAnalisis.mensajecompleto2=mensajecompleto2;
end
ResultadosAnalisis.cor(j,i)=coract;
ResultadosAnalisis.sec(j,i)=toc(inicioiteracion);
% Función que secciona la imagen en bloques
function Matrices=BloquesAnalisisv1(Matrices,alturabloque,anchurabloque)
disp([13,'************SECCIONAR: ,',num2str(alturabloque),'x',num2str(anchurabloque),',************'])
% Defino el numero de secciones y creo la Matriz 4D y una variable para guardar las secciones}
MatrizImagenSteg=Matrices.MatrizImagenSteg;
[alto,ancho,fondo]=size(MatrizImagenSteg);
sec_ancho=floor(ancho/anchurabloque); sec_alto=floor(alto/alturabloque);sec_fondo=fondo;
Matrices.sec_alto=sec_alto; Matrices.sec_ancho=sec_ancho; % Lo guardo para usarlo
clear Matrices.MatrizSeccionesVar;
clear Matrices.MatrizSobrante;
```

```
MatrizSeccionesVar=zeros(sec_alto, sec_ancho,sec_fondo, anchurabloque*alturabloque); % De forma lineal para cada seccion
S_secj_seci_seck=zeros(alturabloque,anchurabloque,3);
% Inicio el bucle que recorre las la imagen y secciono en alturabloque*anchurabloque
\begin{verbatim}
for seck=1:sec_fondo
    for secj=1:sec_alto
        for seci=1:sec_ancho
% Consigo la matriz del bloque para cada color
            S_secj_seci_seck(:,:,seck)=MatrizImagenSteg(((secj-1)*alturabloque)+1:((secj-1)*alturabloque)
            +alturabloque,((seci-1)*anchurabloque)+1:((seci-1)*anchurabloque)+anchurabloque,seck);
% Vectorizo el bloque para que se pueda aplicar fft
            for j=1:alturabloque
                for i=1:anchurabloque
                    % El problema esta aquí al multiplicar por 8
                    MatrizSeccionesVar(secj,seci,seck,((j-1)*anchurabloque)+i)=S_secj_seci_seck(j,i,seck);
                    % Recorre horizontalmente
                end
            end
        end
    end
end
% Añado la parte de imagen sobrante
MatrizSobrante=zeros(size(MatrizImagenSteg));
MatrizSobrante(:,(sec_ancho*anchurabloque)+1:ancho,:)=MatrizImagenSteg(:,(sec_ancho*anchurabloque)+1:ancho,:); % Lado derecho
MatrizSobrante((sec_alto*alturabloque)+1:alto,:,:)=MatrizImagenSteg((sec_alto*alturabloque)+1:alto,:,:); % Abajo
% Compruebo que se ha seccionado bien
Matrices.MatrizSobrante=MatrizSobrante;
Matrices.MatrizSeccionesVar=MatrizSeccionesVar;
[y,x,z,v]=size(MatrizSeccionesVar);
total1=sum(sum(sum(sum(MatrizSeccionesVar))));
total2=sum(sum(sum(MatrizSobrante)));
Matrices.SumaEstegSec=total1+total2;
disp(['Matriz secciones creada. Dimensiones: ',num2str(x),'*',num2str(y),'*',num2str(z),'.vector',num2str(v),'.']);
if(Matrices.SumaEsteg==Matrices.SumaEstegSec)
    disp('Bien Seccionado')
else
    diferencia=abs(Matrices.SumaEsteg-Matrices.SumaEstegSec);
    disp(['Mal Seccionado, diferencia suma: ',num2str(diferencia)])

end
```

A.3. Código Octave del sistema de esteganoanálisis para ejecutar en cloud computing

La comunicación con Octave se realiza a través de un terminal, por ello no se muestran imágenes en este código.

1. Análisis de una iteración en Matlab.

2. Función que secciona la imagen.

1. Función que calcula cada iteración

```
function corre=AnalisisAmazonv1(j,i)
% Los resultados por TERMINAL o FICHEROS, no usar imágenes
tic;
% Se carga la imagen
path='/home/'; % Los archivos se transfieren al fichero home de la máquina de Amazon
load([path,'Matrices.mat']);
% Funcion de probabilidad del español antiguo / ingles
VectorEsp=[12.2 1.5 3.6 5.3 14 0.5 1 1.2 5.5 0.6 0 5.4 2.7 6.9 9.9 2.2 2 6.2 7.7 3.8 4.8 1.1 0 0 1.5 0.4]; %Don quijote
%vectorhist=[12.53 1.42 4.68 5.68 13.68 0.69 1.01 0.7 6.25 0.44 0.01 4.97 3.15 6.71 8.68 2.51 0.88 6.87
 7.98 4.63 3.93 0.9 0.02 0.22 0.9 0.52]; %Espanol
```

```
% Coloco la función de probabilidad en su lugar ASCII
Matrices.HistoEspPorc=zeros(1,128);
Matrices.HistoEspPorc(98:123)=VectorEsp(1:26)/100;
m1=max(Matrices.HistoEspPorc(97:122));
% Divido la imagen en Secciones de j*i
inicioiteracion=tic;
% Llamada a la función BloquesAnalisis que secciona la imagen
Matrices=BloquesAnalisisv1(Matrices,j,i);
% Análisis en bloques i*j el numero de secciones en bloque analisis
disp([13,'************ANALISIS************'])
mensajecompleto2='';
for seck=1:3
       for secj=1:Matrices.sec_alto
        for seci=1:Matrices.sec_ancho
% Trasformada de Fourier del vector de la seccion}
            freq_sec_analysis_lineal=fft(Matrices.MatrizSeccionesVar(secj,seci,seck,:)); % Discrete fourier transform
            % OCTAVE SOLO ACEPTA INTEGERS en dec2bin HAY QUE HACER UN ROUND!!
            freq_sec_analysis_lineal_bin1=dec2bin(65536+round(real(freq_sec_analysis_lineal))); % Parte real: +65536 offset
            % freq_sec_analysis_lineal_bin2=dec2bin(65536+imag(freq_sec_analysis_lineal));
      % Obtengo el mensaje esteganografio con el bit menos significativo}
            [filas,columnas]=size(freq_sec_analysis_lineal_bin1);
            mensaje2_bin2(1:7)=freq_sec_analysis_lineal_bin1(1:7,columnas);
            mensaje2=bin2dec(mensaje2_bin2);
            mensajechar=num2str(mensaje2);
            mensajecompleto2=[mensajecompleto2,mensajechar];
        end
        mensajecompleto2=[mensajecompleto2,13];
    end
end
mensajecompleto2;
toc
% Normalizo los histogramas para hacer la correlacion
HistoMensaje=hist(abs(mensajecompleto2),128); % Comparare resultados con lenguaje, 128 ASCII
HistoMensajePorc=HistoMensaje/length(mensajecompleto2); m2=max(HistoMensajePorc(97:128));
HistoMensajeNorm=HistoMensajePorc*m1/m2; % Normalizo
% Hallo la correlación
% corr2 en Matlab => cor en Octave core
coract=cor( HistoMensajeNorm(97:122),Matrices.HistoEspPorc(97:122));
disp(['La correlacion es: ',num2str(coract)]);
% devulevo la correlación
corr=coract
```

2. Función que secciona la imagen

```
function Matrices=BloquesAnalisisv1(Matrices,alturabloque,anchurabloque)
disp([13,'************SECCIONADO EN: ,',num2str(alturabloque),'x',num2str(anchurabloque),',************'])
% Defino el numero de secciones y creo la Matriz 4D y una variable para guardar las secciones
MatrizImagenSteg=Matrices.MatrizImagenSteg;
[alto,ancho,fondo]=size(MatrizImagenSteg);
sec_ancho=floor(ancho/anchurabloque); sec_alto=floor(alto/alturabloque);sec_fondo=fondo;
Matrices.sec_alto=sec_alto; Matrices.sec_ancho=sec_ancho; % Lo guardo para usarlo
%clear Matrices.MatrizSeccionesVar;
%clear Matrices.MatrizSobrante;
MatrizSeccionesVar=zeros(sec_alto, sec_ancho,sec_fondo, anchurabloque*alturabloque); % De forma lineal para cada seccion
S_secj_seci_seck=zeros(alturabloque,anchurabloque,3);
% Inicio el bucle que recorre las la imagen y secciono en alturabloque*anchurabloque
for seck=1:sec_fondo
    for secj=1:sec_alto
        for seci=1:sec_ancho
% Consigo la matriz del bloque para cada color
            S_secj_seci_seck(:,:,seck)=MatrizImagenSteg(((secj-1)*alturabloque)+1:((secj-1)*alturabloque)
            +alturabloque,((seci-1)*anchurabloque)+1:((seci-1)*anchurabloque)+anchurabloque,seck);
            %whos
% Vectorizo el bloque para que se pueda aplicar fft}
```

```
            for j=1:alturabloque
                for i=1:anchurabloque
                    % El problema esta aquí al multiplicar por 8
                    MatrizSeccionesVar(secj,seci,seck,((j-1)*anchurabloque)+i)=S_secj_seci_seck(j,i,seck);
                     % Recorre horizontalmente
                end
            end
        end
    end
end
% Añado la parte de imagen sobrante
MatrizSobrante=zeros(size(MatrizImagenSteg));
MatrizSobrante(:,(sec_ancho*anchurabloque)+1:ancho,:)=MatrizImagenSteg(:,(sec_ancho*anchurabloque)+1:ancho,:); % Lado derecho
MatrizSobrante((sec_alto*alturabloque)+1:alto,:,:)=MatrizImagenSteg((sec_alto*alturabloque)+1:alto,:,:); % Abajo
%Compruebo que se ha seccionado bien
Matrices.MatrizSobrante=MatrizSobrante;
Matrices.MatrizSeccionesVar=MatrizSeccionesVar;
[y,x,z,v]=size(MatrizSeccionesVar);
total1=sum(sum(sum(sum(MatrizSeccionesVar))));
total2=sum(sum(sum(MatrizSobrante)));
Matrices.SumaEstegSec=total1+total2;
```